Programozási

III.

Nemes Tihamér
Nemzetközi Informatikai Tanulmányi Verseny
2000-2004

Szerkesztette: Zsakó László

A Nemes Tihamér NITV feladatsorait

Hanák Péter (BME)
Horváth Gyula (SzTE)
Zsakó László (ELTE)

vezetésével az NJSzT Országos Versenybizottsága mindenkori tagjai dolgozták ki:

Gulyás László (ELTE), Horváth László (ELTE), Papp Zoltán (DE), Szabadhegyi Csaba (ELTE), Szlávi Péter (ELTE), Marhefka István (BME), Rácz Balázs (BME), Tóth László (BME), Újhelyi Gábor (BME).

Ezen túl a BME és az ELTE számos hallgatója vett részt a feladatok kidolgozásában és a verseny lebonyolításában.

Nem látó (vak) tanulók számára a feladatszövegeket Braile-írással

Helfenbein Henrik (ELTE)

állította elő.

A kiadvány a Neumann János Számítógép-tudományi Társaság által kiadott *Programozási versenyfeladatok tára 3* című kötet alapján készült.

ISBN 978-963-489-065-2

A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2018-ban.

Eötvös Loránd Tudományegyetem
Informatikai Kar

Előszó

Magyarországon több kezdeti próbálkozás után 1985-ben szervezte meg az első országos középiskolai programozási versenyt Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Verseny (NTOKSzTV) néven a Neumann János Számítógép-tudományi Társaság (NJSZT). A 9 évig kétfordulós versenyen 1989-ig egy korcsoportban, 1990-től külön korcsoportban indulhatnak a 14-15, illetve a 16-19 éves tanulók (azaz a középiskolák I.-II., illetve III.-V. osztályos diákjai). Az 1993/94-es tanévtől a versenyt három fordulóban rendezzük (az iskolai forduló és az országos döntő közé iktattunk egy regionális-megyei fordulót). A verseny első tíz helyezettje – a többi országos középiskolai tanulmányi versenyhez hasonlóan – érettségi, illetve felvételi kedvezményben részesül, és közülük válogatjuk ki az 1989 óta ugyancsak évente megrendezett Nemzetközi Informatikai Diákolimpia magyar résztvevőit is.

A Nemes Tihamér verseny, szerénytelenség nélkül megállapíthatjuk, népszerűvé vált a diákok körében: az utóbbi években 260-300 magyarországi középiskola kb. 2500-2500 I.-II., illetve III.-V. osztályos diákja vett részt a verseny első, iskolai fordulójában. Közülük kb. 600 diák jutott tovább a második fordulóba, majd kb. 180 a harmadikba. Említést érdemel, hogy a környező országokban élő, magyar anyanyelvű vagy magyarul jól beszélő diákok közül is egyre többen érdeklődnek a verseny iránt, illetve vesznek részt a versenyen; az Erdélyi Magyar Műszaki Tudományos Társaság szervezésében évente kb. 500 diák indul.

1991 óta különböző szervezési formákban az általános iskolások is részt vehetnek országos programozási jellegű versenyen. A határon túli résztvevők, valamint az általános iskolások megjelenése miatt a versenyt átkereszteltük, új neve: Nemes Tihamér Nemzetközi Informatikai Tanulmányi Verseny.

Sokak igényét elégítettük ki azzal, hogy a Nemes Tihamér NITV-t e korosztállyal bővítettük, a megrendezésre jelentkező megyéket (regionális versenybizottságokat) feladatokkal láttuk el, s megszerveztük az országos döntőt is.

Az NJSZT Országos Versenybizottsága sokak kívánságát teljesíti most azzal, hogy önálló kötetekben megjelenteti az eddigi versenyfeladatokat. Reméljük, hogy ezzel is segíteni tudjuk a leendő versenyzőket a felkészülésben, a tanáraikat pedig a számítástechnikai foglalkozások és szakkörök megtartásában.

Ebben a kötetben a 2000-2004 közötti Nemes Tihamér versenyek programozási feladatait ismertetjük. Minden egyes feladat után a javító tanároknak szóló megoldási és értékelési útmutatót is közöljük. A példatárban egyedülálló módon a feladatok részletes megoldását is közöljük, amelyek eddig még sehol nem jelentek meg. A versenyekre készülő diákoknak természetesen azt javasoljuk, hogy mielőtt a közölt megoldást és értékelést elolvasnák, saját maguk, önállóan próbálják megoldani a kitűzött feladatot. Nem törekedtünk szöveghűségre: ahol az eredeti megfogalmazást pontatlannak vagy hibásnak találtuk, módosítottunk a szövegen.

Tartalom

Nemes Tihamér
Nemzetközi Informatikai Tanulmányi Verseny

2000. Első forduló

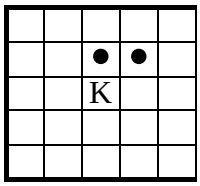
Ötödik-nyolcadik osztályosok

1. feladat: Karesz a robot (30 pont)

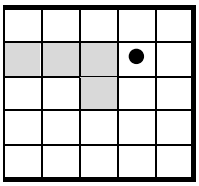
Karesz, a robot egy egyszerű négyzetrácsos világban él. A zsebében köveket hordhat, amelyeket séta közben a négyzetrács mezőiről szedhet fel. Egy-egy követ az ábrákon egy-egy fekete pötty jelöl. Kezdetben a terület közepén, az ábrán K-val jelölt helyen áll, északi irányba néz és nincs nála kő.

```
Eljárás:
  Ismételd amíg nem érsz ki
    Lépj előre
    Ha van ott kő akkor Vegyél fel egyet
                        Fordulj balra
    Elágazás vége
  Ismétlés vége
Eljárás vége.
```

Példa:



Karesz a 3. lépése után ér ki. (a 4.-re már lelépne) Kilépéskor 1 kő van a zsebében. A kilépésig a szürke mezőkön jár.



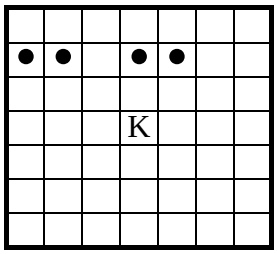
Az alábbi kezdőállapotok esetén add meg, hogy

1. hány lépés után lép ki Karesz a négyzetrácsról,

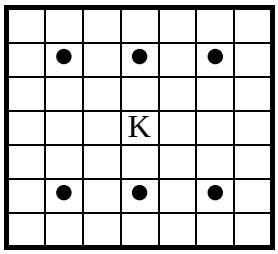
2. kilépéskor hány kő van a zsebében,

3. mely mezőkön jár a kilépésig.

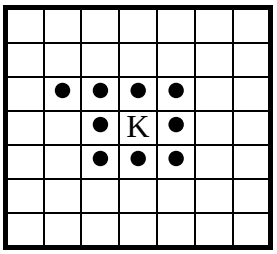
A.



B.



C.



2. feladat: Falfestés (24 pont)

Egy festő egy falcsíkot szeretne balról jobbra haladva simára festeni. Egy menetben minden egyes szakaszt 1 vagy 2 vastagságú rétegben fest az alábbi algoritmusrészlet alapján. Az algoritmusban `FAL(I)` jelöli, hogy kezdetben az I. szakasz (1≤I≤N) milyen vastagságú, s az eredmény is itt keletkezik.

```
Ciklus I=2-től N-ig
  Ha FAL(I)=FAL(I-1)-1 akkor FAL(I):=FAL(I)+1
  különben ha FAL(I)<FAL(I-1)-1 akkor FAL(I):=FAL(I)+2
Ciklus vége
```

A. Az alábbi kezdő vastagságok esetén válaszolj a kérdésekre:

```
1) 5,1,2,1,3,1    2) 1,2,3,5,1,1      3) 1,5,2,7,2
```

Aa. Hány menet után nem tud tovább festeni?

Ab. Az egyes menetek után milyen vastag lesz a fal az egyes helyeken?

B. Mi a kezdeti feltétele annak, hogy a festés befejeztével sima legyen a fal?

3. feladat: Válassz ki kettőt (26 pont)

Az alábbi három algoritmus kis eltéréssel majdnem egyforma. Mindegyik 100 számot olvas be, melyek 0-nál nagyobbak és 1000-nél kisebbek, s ezek közül ad meg 2 számot.

```
Első:
  A:=0; B:=1000
  Ciklus 100-szor
    Be: C
    Ha C>A akkor A:=C különben Ha C<B akkor B:=C
  Ciklus vége
Eljárás vége.
```

```
Második:
  A:=0; B:=0
  Ciklus 100-szor
    Be: C
    Ha C>A akkor B:=A; A:=C különben Ha C>B akkor B:=C
  Ciklus vége
Eljárás vége.
```

```
Harmadik:
  A:=0; B:=0
  Ciklus 100-szor
    Be: C
    Ha C>A akkor B:=A; A:=C
        különben Ha C<A és C>B akkor B:=C
  Ciklus vége
Eljárás vége.
```

A. Mi lesz a három algoritmus végén az A és a B változók tartalma?

B. Melyeknél lehetséges, hogy B ciklus előtti értéke megmarad, s mi ennek a feltétele?

4. feladat: Ládapakoló robot (20 pont)

Egy robot egy raktárban ládákat pakol a polcokra. A polcok egymás mellett vannak. A robot kezdetben a bal oldalinál áll, új ládát csak itt foghat a kezébe, jobbra és balra lépkedhet, s a kezében levő ládát az előtte levő polcra teheti. Minden polcra tetszőleges darabszámú ládát tehet, de kisebbre nagyobbat nem (a ládák mérete 1 és 999 közötti). Üres polc esetén a legfelső láda 1000 méretű.

Az alábbi két algoritmus alapján pakolhat egy ládát valamelyik polcra (a robot az eljárások elején mindig automatikusan a bal oldali polchoz áll):

```
Első:
  Vedd fel a ládát
  Ismételd amíg a láda nagyobb, mint az előtted levő polc
                legfelső ládája és polc előtt állsz
    Lépj egyet jobbra
  Ismétlés vége
  Tedd a polcra a ládát
Eljárás vége.
```

```
Második:
  Vedd fel a ládát
  Ismételd amíg a láda nem nagyobb, mint az előtted levő
                polc legfelső ládája és polc előtt állsz
    Lépj egyet jobbra
  Ismétlés vége
  Lépj egyet balra; Tedd a polcra a ládát
Eljárás vége.
```

Tegyük fel, hogy a raktárban 3 polc van, s N láda érkezik az alábbi méretekkel: 5,8,4,9,1,3,6.

A. Az egyes eljárások alapján melyik ládát hova teszi a robot?

B. Fogalmazd meg, hogy mikor nem tudja a robot elhelyezni a következő ládát!

C. Add meg, hogy a fenti ládasorozatot milyen méretű ládával kell kibővíteni, hogy a feladat ne legyen megoldható!

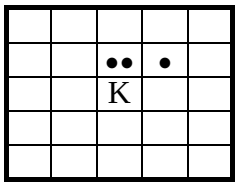
Kilencedik-tizedik osztályosok

1. feladat: Karesz a robot (21 pont)

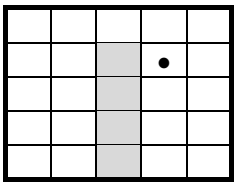
Karesz, a robot egy egyszerű négyzetrácsos világban él. A zsebében köveket hordhat, amelyeket séta közben a négyzetrács mezőiről szedhet fel. Egy-egy követ az ábrákon egy-egy fekete pötty jelöl. Kezdetben a terület közepén, az ábrán K-val jelölt helyen áll, északi irányba néz és nincs nála kő.

```
Eljárás:
  Ismételd amíg nem érsz ki
     Ha van ott kő akkor Vegyél fel egyet; Fordulj balra
                    különben Lépj előre
     Elágazás vége
  Ismétlés vége
Eljárás vége.
```

Példa:



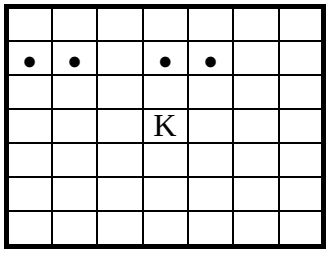
Karesz a 4. lépése után ér ki (az 5.-re már lelépne). Kilépéskor 2 kő van a zsebében. A kilépésig a szürke mezőkön jár.



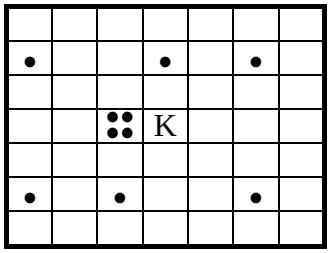
Az alábbi kezdőállapotok esetén add meg, hogy

1. hány lépés (a `Lépj előre` utasítás végrehajtásai száma) után lép le Karesz a négyzetrácsról,
2. kilépéskor hány kő van a zsebében,
3. mely mezőkön jár a kilépésig.

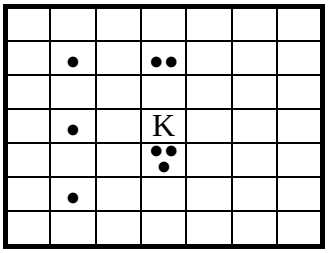
A.



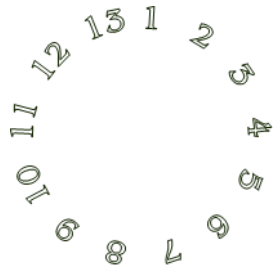
B.



C.



2. feladat: Kiszámolós (20 pont)



Egy kiszámolós játékban N gyerek körbe áll az ábrának megfelelően:

Az alábbi három algoritmus különféle kiszámolási szabály szerint működik. Kezdetben az A logikai vektor összes eleme (1-től N-ig) igaz értékű.

A. Add meg, milyen sorrendben írja ki a három algoritmus a kiszámolt gyerekek sorszámát, ha kezdetben N=13 és K=5?

B. Melyek kerülhetnek végtelen ciklusba, s mi ennek a feltétele?

```
Első:
  I:=K; A(I):=hamis; Ki(I); J:=1
  Ciklus amíg J<N
    I:=I+K; Ha I>N akkor I:=I-N
    Ha A(I) akkor A(I):=hamis; Ki(I); J:=J+1
  Ciklus vége
Eljárás vége.
```

```
Második:
  I:=K; A(I):=hamis; Ki(I); J:=1
  Ciklus amíg J<N
    L:=0
    Ciklus amíg L<K
      I:=(I mod N)+1
      Ha A(I) akkor L:=L+1
    Ciklus vége
    A(I):=hamis; Ki(I); J:=J+1
  Ciklus vége
Eljárás vége.
```

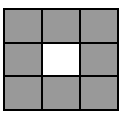
```
Harmadik:
  I:=K; A(I):=hamis; Ki(I); J:=1; Ha K>1 akkor K:=K-1
  Ciklus amíg J<N
    L:=0
    Ciklus amíg L<K
      I:=I-1; Ha I=0 akkor I:=N
      Ha A(I) akkor L:=L+1
    Ciklus vége
    A(I):=hamis; Ki(I); J:=J+1; Ha K>1 akkor K:=K-1
  Ciklus vége
Eljárás vége.
```

3. feladat: Grafikus utasítások (18 pont)

Egy rajzgépet a `RAJZ S` utasítással lehet vezérelni, ahol `S` szöveg típusú változó. A szövegben grafikus utasítások szerepelhetnek (E,K,D,N – észak, kelet, dél, nyugat). A megfelelő betű hatására a rajzgép tolla 1 egységgel elmozdul (és így rajzol) az adott irányba.

Példa:

RAJZ "EEKKDDNN" a négyzetrács bal alsó sarkából indulva az alábbi képpontokat festi be:



A. Mit rajzol a `RAJZ "EEEKKDKKKKEKKDDDNNNDNNENNN"` utasítás hatására?

B. Miben más a `RAJZ f(S)` utasítás hatására készülő rajz, mint a `RAJZ S`, ha az `f` függvény az alábbi?

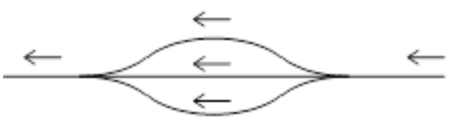
```
B1. f(S):
      Ciklus I=1-től hossz(S)-ig
        Ha S(I)="E" akkor T(I):="D"
        különben ha S(I)="D" akkor T(I):="E"
        különben T(I):=S(I)
      Ciklus vége
      f:=T
    Függvény vége.
```

```
B2. f(S):
      Ciklus I=1-től hossz(S)-ig
        Ha S(I)="E" akkor T(I):="K"
        különben ha S(I)="K" akkor T(I):="D"
        különben ha S(I)="D" akkor T(I):="N"
        különben T(I):="E"
      Ciklus vége
      f:=T
    Függvény vége.
```

4. feladat: Rendezőpályaudvar (21 pont)

Egy rendező-pályaudvaron K sínpárra tudják tolni a vagonokat. Egy hosszú szerelvény érkezik, amelyben N-féle helyre menő vagon van. Egy sínpárra egyszerre csak ugyanarra a helyre menő vagonokat tolhatunk, s onnan visszatolni már nem szabad. Ha egy sínpáron olyan helyre küldendő vagonok állnak, amelyek az érkező szerelvény további részében már nincsenek, akkor azokat el kell indítani a célállomás felé, és a sínpár másik szerelvény összeállításához használható. Az egyes vagonokat a célállomás neve azonosítja.

Példa:



Ha 3 sínpár van, az érkező szerelvény az `a,b,a,b,c,a,c` helyre küldendő vagonokból áll, akkor 2 sínpárral megoldható a feladat. Az elsőre toljuk az `a` célállomásra menőket, a másodikra előbb a `b`-re menőket, s amikor a 4. kocsival végeztünk, a szerelvényt a második vágányról elindíthatjuk, s a helyére most már a `c` állomásra menők kerülhetnek. Így a két vágányra kerülő vagonok sorszámai:

1: `1,3,6` **2:** `2,4,5,7`

A. Mi a feltétele, hogy minden célállomásra egyetlen szerelvényt kell küldeni?

B. Hány sínpárt kell minimálisan használni, ha minden célállomásra egyetlen szerelvényt kell küldeni?

C. Add meg a minimálisan használandó sínpárok számát, valamint hogy az egyes sínpárokra az eredeti szerelvény milyen sorszámú kocsijait kell tolni az alábbi két példára (a megoldáshoz legfeljebb 3 sínpárt használhatsz):

`1. a,a,b,a,c,a,d,a,b,d,b,d` `2. a,a,b,a,c,a,d,c,d,e,e,d`

5. feladat: Szavak sorrendje (20 pont)

Egy rekurzív programozási nyelvben egy mondat szavaira az alábbi függvényeket alkalmazhatjuk:

`szószám(Mondat)`: a mondat szavai számát adja

`első(Mondat)`: a mondat első szavát adja

`utolsó(Mondat)`: a mondat utolsó szavát adja

`elsőutániak(Mondat)`: a mondat összes szavát adja a másodiktól az utolsóig

`utolsóelőttiek(Mondat)`: a mondat összes szavát adja az utolsót elhagyva

`elejére(Szó,Mondat)`: a szót a mondat elejére illeszti, s ez lesz az eredmény

`végére(Szó,Mondat)`: a szót a mondat végére illeszti, s ez lesz az eredmény

Add meg, hogy az alábbi rekurzív függvények milyen sorrendben írják ki a "EGY KETTŐ HÁROM NÉGY ÖT HAT HÉT NYOLC KILENC TÍZ" mondat szavait!

```
A(M):
  Ha szószám(M)>0 akkor végére(első(M),A(elsőutániak(M)))
  különben M
Függvény vége.
```

```
B(M):
  Ha szószám(M)>2 akkor
     elejére(első(M),elejére(utolsó(M),
                        B(elsőutániak(utolsóelőttiek(M)))))
  különben M
Függvény vége.
```

```
C(M):
  Ha szószám(M)>1 akkor
     elejére(első(M),C(elsőutániak(elsőutániak(M))))
  különben M
Függvény vége.
```

```
D(M):
  Ha szószám(M)>1 akkor
     végére(első(M),elejére(első(M),D(elsőutániak(M))))
  különben M
Függvény vége.
```

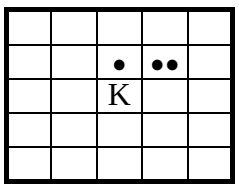
Tizenegyedik-tizenharmadik osztályosok

1. feladat: Karesz a robot (18 pont)

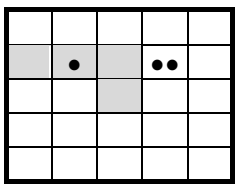
Karesz, a robot egy egyszerű világban él, melyet egy négyzetráccsal lehet leírni. A zsebében köveket hordhat, amelyeket letehet a négyzetrács mezőire, illetve felveheti az ott talált követ. Kezdetben a terület közepén (az ábrán K-val jelölt helyen) áll, északi irányba néz és nincs nála kő. (A köveket az ábrákon fekete pöttyök jelzik.)

```
Eljárás:
  Ismételd amíg nem érsz ki
     Ha van nálad kő, akkor Tegyél le egyet
     Lépj előre
     Ha van ott kő akkor Vegyél fel egyet
                         Fordulj balra
                         Lépj előre
     Elágazás vége
  Ismétlés vége
Eljárás vége.
```

Példa:



Karesz a 3. lépése után ér ki (a 4.-re már lelépne). A kilépésig a szürke mezőkön jár. A köveket a jelzett helyen hagyja.



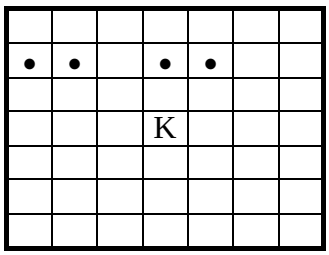
Az alábbi kezdőállapotok esetén add meg, hogy

1. hány lépés (a `Lépj előre` utasítás végrehajtásai száma) után lép le Karesz a négyzetrácsról,

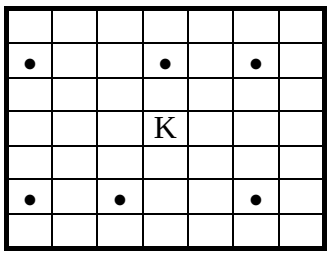
2. mely mezőkön jár a kilépésig,

3. hol maradnak kövek?

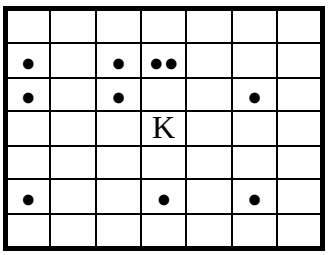
A.



B.

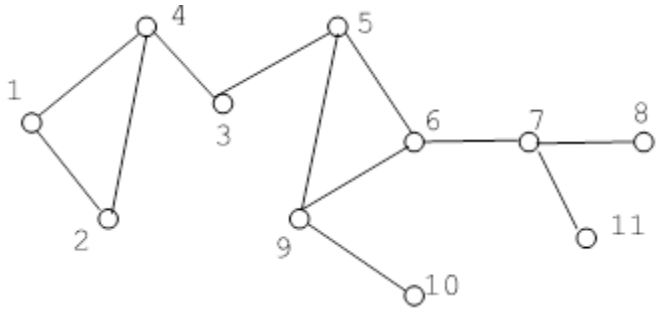


C.



2. feladat: Gráfalgoritmus (20 pont)

Egy irányítatlan gráfot a G(N,N) csúcsmátrix-szal ábrázolunk, azaz G(I,J) pontosan akkor IGAZ értékű, ha a gráfban az I. és a J. csúcsot él köti össze. Két tetszőleges csúcs között legfeljebb egy él lehet, önmagával egyetlen csúcsot sem köt össze él. Az alábbi algoritmus bizonyos éleket töröl a gráfból. Az S vektor elemei értéke kezdetben 0.




```
Törlés:
  Ciklus I=1-től N-1-ig
    Ciklus J=I+1-től N-ig
      Ha G(I,J) akkor S(I):=S(I)+1; S(J):=S(J)+1
    Ciklus vége
  Ciklus vége                                        (*)
  Ciklus
    I:=1
    Ciklus amíg I≤N és S(I)≠1
      I:=I+1
    Ciklus vége
    Ha I≤N akkor S(I):=0                             (**)
                 J:=1
                 Ciklus amíg nem G(I,J)
                   J:=J+1
                 Ciklus vége
                 S(J):=S(J)-1; G(I,J):=hamis; G(J,I):=hamis
    Elágazás vége
  amíg I≤N
  Ciklus vége
Eljárás vége.
```

A. Mit tartalmaz az S vektor a (*)-gal jelölt helyen? Add meg a konkrét példa esetén is!

B. Milyen csúcsok esetén jutunk el a (**)-gal jelölt részre? Add meg a konkrét példa esetén is!

C. Milyen élek maradnak végül a gráfban? Add meg a konkrét példa esetén is!

3. feladat: Szöveggyártás (15 pont)

Az alábbi algoritmusok az A vektort használják bemenetként. A vektor elemeit az angol ábécé kisbetűivel indexeljük, értékük nemnegatív egész szám.

```
Kiszámol(A,N):
  N:=0
  Ciklus J='a'-tól 'z'-ig
    Ha A(J)>0 akkor N:=N+A(J)
  Ciklus vége
Eljárás vége.
```

```
Mitcsinál(I,N,A,B):
  Ha I=N+1 akkor Ki(B,N)                    (*)
  különben
    Ciklus J='a'-tól 'z'-ig
      Ha A(J)>0 akkor B(I):=J; A(J):=A(J)-1
                      Mitcsinál(I+1,N,A,B); A(J):=A(J)+1
      Elágazás vége
    Ciklus vége
  Elágazás vége
Eljárás vége.
```

Előbb a `Kiszámol(A,N)`, utána a `Mitcsinál(1,N,A,B)` eljárást hajtjuk végre. A `Ki(B,N)` a B vektor elemeit írja ki 1-től N -ig. Az alábbi kérdésekre szöveges választ adj!

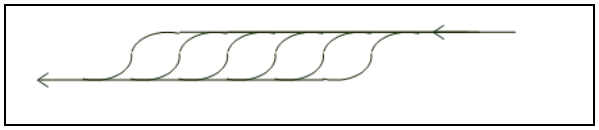
A. Mit ír ki a (*)-gal jelölt sorban, amikor a kiírást először végrehajtjuk (hogyan függ a kiírás az A vektor tartalmától)?

B. Hányszor hajtja végre a (*)-gal jelölt sorban levő kiírást?

C. A (*)-gal jelölt sorban levő kiírások eredményei milyen sorrendben követik egymást?

4. feladat: Prioritási sor (21 pont)

A prioritási sor olyan adatszerkezet, amelybe új elemeket a fontosságuknak megfelelő helyre lehet beilleszteni (a kisebb értékűek előbbre kerülnek, mint a nagyobbak), a sort elhagyni azonban csak a sor elején álló tudja. A prioritási sort jelképezi a mellékelt ábra.



Egy K (1<K≤N) hosszúságú prioritási sort használunk az N elemű A vektor rendezésére az alábbi módon:

```
Rendezés:
  Ciklus I=1-től K-ig
    Sorba(A(I))
  Ciklus vége
  Ciklus I=K+1-től N-ig
    Sorból(B(I-K)); Sorba(A(I))
  Ciklus vége
  Ciklus I=1-től K-ig
    Sorból(B(N-K+I))
  Ciklus vége
  A:=B
Eljárás vége.
```

Legyen kezdetben N=8, K=4 és az A vektor a következő: 5,4,3,6,8,5,9,7.

A. Írd le ciklusléptésenként a prioritási sor tartalmát (mindhárom ciklusra)!

B. Mi a feltétele általában annak, hogy az eredmény valóban rendezett legyen?

C1. Hányszor kell végrehajtani ezt az algoritmust egy tetszőleges kiinduló sorozaton, hogy biztosan rendezett legyen a végeredmény?

C2. Milyen bemenetre kell valóban ennyiszer végrehajtani?

5. feladat: Fraktál (26 pont)

Rekurzív ábrák (ún. fraktálok) generálását egy rekurzív formulapárral (szövegesen) adhatjuk meg. A formula egyik tagja (az ún. axióma) megadja az alapábrát. a másik pedig a helyettesítési szabályokat, amit az első tagra kell alkalmazni, az előírt darabszámszor.

A formulákban rajzoláskor az F betű 1 egységnyi rajzolást jelent az aktuális irányban, az X betű hatására semmit nem kell csinálni, a + balra fordulást, a – jobbra fordulást jelent (ennek szögét az axiómákban adjuk meg).

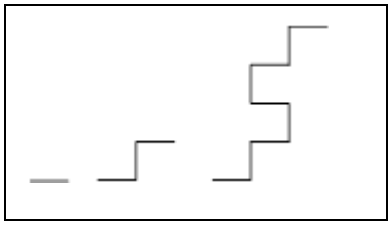
Példa:

Axióma: F szög (+,–): 90 fok helyettesítési szabály: F=F+F–F

Az első helyettesítés után az axiómából keletkezik: F+F–F.

A második helyettesítés után az előzőből keletkezik:

F+F-F+F+F–F–F+F–F.



A kezdőábra, valamint a helyettesítések után kapott ábrák:

Az alábbi két formulapárhoz

A. rajzold le az alapábrát;

B. add meg, hogy a kiinduló ábrát leíró szöveg hogyan változik, ha a helyettesítési szabályokat egyszer, illetve kétszer alkalmazzuk;

C. valamint rajzold le az így keletkező, a generálásnak megfelelő ábrákat!

1. axióma: F--F--F

szög (+,-): 60 fok

helyettesítési szabályok:

F=F+F--F+F

2. axióma: FXF

szög (+,-): 120 fok

helyettesítési szabályok:

F=FXF

X=+FXF-FXF-FXF+

2000. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Tükördátumok (25 pont)

Az 1901 és 1999 közötti dátumokat a EE.HH.NN alakban írja számítógépünk. Az évszámból az utolsó két jegyet írja ki. Ha az így kapott dátumban az év, a hónap vagy a nap 0-val kezdődne, akkor egyetlen számjeggyel kell kiírni. Tükördátumnak nevezzük azokat a dátumokat, amelyek számjegyei sorrendjét megfordítva is dátumot kapunk.

Készíts programot, amely beolvas egy évszámot (1900<évszám<2000), majd naptári sorrendben kiírja ebben az évben a tükördátumokat!

Példa:

```
Bemenet: 1982
Kimenet: 82.1.28, 82.2.8, 82.2.28, 82.3.28, 82.4.28, 82.5.28,
82.6.28, 82.7.28, 82.8.28, 82.9.28, 82.11.28, 82.12.8
```

2. feladat: Memóriajáték (24 pont)

Egy memóriajátékot két játékos játszik. Az egyik mond egy karaktersorozatot (X), s a másiknak azt ugyanabban a sorrendben kell megismételnie (Y). A második játékos karaktereket hagyhat ki az eredeti sorozatból, a sorrendet azonban nem változtathatja meg.

Készíts programot, amely megadja hogy

A. a második játékos meddig tudta pontosan ismételni az első által mondott karakter-sorozatot!

B. a második játékos szavának azt a legnagyobb kezdőszeletét, amely a kihagyási szabálynak megfelel!

Példa:

```
bemenet:
     X: ABCDEFGH
     Y: ABDFCX
eredmény:
     A feladat: AB
     B feladat: ABDF
```

3. feladat: Programozási verseny (26 pont)

Egy programozási versenyen minden versenyző választhat egy programozási nyelvet, amin dolgozni fog. Programodnak először a versenyzők N (≤20) számát kell beolvasnia, majd pedig a versenyzők nevét, illetve a választott programozási nyelvet!

Példa:

```
3
Kiss Péter
Pascal
Nagy Tamás
Logo
Kovács Anna
Pascal
```

Készíts programot, amely:

A. A képernyőre írja a használt nyelvek számát.

Példa:

```
    2 nyelv
```

B. A képernyőre írja, hogy az egyes nyelvekből hány versenyző van.

Példa:

```
    Pascal: 2       Logo: 1
```

C. A képernyőre írja az egyes nyelveken versenyzők nevét.

Példa:

```
    Pascal: Kiss Péter, Kovács Anna
```

```
Logo: Nagy Tamás
```

Kilencedik-tizedik osztályosok

1. feladat: Gombaszedés (22 pont)

Józsi bácsi hétvégénként nagy szeretettel jár a közeli erdőbe gombászni. Valamelyik hátizsákjával szokott elindulni, és ebbe a zsákba próbál meg minél több gombát szedni. Az erdőben háromféle gomba terem meg: Csiperke, Rókagomba és Lila pereszke. Józsi bácsi az évek során a következő módszert alakította ki a gombák leszedésére:

- Mindaddig nem szed Lila pereszkét, amíg még van az erdőben Csiperke gomba.
- Először mindig a legnagyobb (legnehezebb) gombát szedi le (persze, ha ez nem ütközik az A. ponttal).
- Azonos súlyú gombák esetén először Csiperké(ke)t, majd a Rókagombá(ka)t, és legvégül a Lila pereszkéket szedi le.

Ha adott az erdő gombaállománya (minden gomba súlya dekagrammban és fajtája), és Józsi bácsi hátizsákjának kapacitása (azaz hány dekagramm gombát képes a zsákban tárolni) határozzuk meg, hogy hány dekagramm gombát tud (és ezekből fajtánként mennyit) leszedni. Tudjuk azt is, hogy Józsi bácsi nem vág ketté gombát, azért, hogy a zsákja még jobban tele legyen (tehát csak egész gombák vannak a zsákban). Ha egy nagyobb gomba már nem fér bele a zsákba, azt kihagyja.

Készíts programot a gombák zsákba pakolására!

A GOMBA.BE állomány első sorában a gombák száma van (maximum 1000), a másodikban pedig a zsák kapacitása (maximum 100 000), a többi sor mindegyikében egy gomba jellemzői vannak: a fajta (a következő karakterek valamelyike: C, R vagy L) és a súly dekagrammban (1 és 100 között) egy szóközzel elválasztva.

A GOMBA.KI állományba a leszedett gombák darabszámát és súlyát kell írni! Az első sorba az összes leszedett gomba, a másodikban csak a csiperke, a harmadikban a rókagomba, a negyedikben pedig a lila pereszke darabszámát és a súlyát!

Példa:

```
GOMBA.BE
6
25
C 4
R 8
L 12
C 10
L 2
R 2
```

Józsi bá' a következő sorrendben szedné le a gombákat: C 10, R 8, C 4, L 12, R 2, L 2. De sajnos a zsákja pici, ezért csak a C 10, R 8, C 4, R 2 gombák férnek el benne (az L 12 gombát kihagyja, mert a zsákba nem fér bele, így megpróbálja a sorban következőt belerakni).

```
GOMBA.KI
4 24
2 14
2 10
0 0
```

2. feladat: Hálózat (20 pont)

Egy számítógépes hálózat kiépítése a következőképpen történik: Kezdetben egy gépből áll a hálózat. Egy új gép bekapcsolásakor azt pontosan 1 db, már a hálózatban levő géppel kötik össze, ami kétirányú kapcsolatot biztosít a két összekötött gép között.

Két gép távolságán a legkevesebb közvetlen összekötést tartalmazó összekötést értjük. A hálózat átmérőjének nevezzük a hálózatban levő gépek közül a két legtávolabbi távolságát.

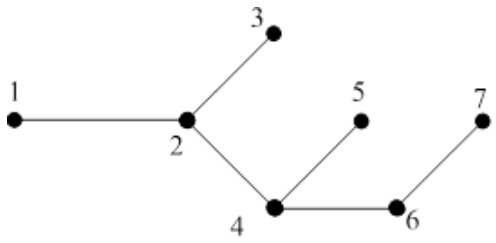
Készíts programot, ami kiszámítja, hogy N db gép esetén mekkora lesz a hálózat átmérője!

A HALOZAT.BE állomány első sorában a gépek száma (1≤N≤100) található. Az állomány I. sorában annak a számítógépnek a J (J<I) sorszáma van, amelyhez az I. számítógépet kötik a hálózatban.

A HALOZAT.KI állományba egyetlen számot kell írni, a hálózat átmérőjét a teljes kiépülés után!

Példa:

```
HALOZAT.BE      HALOZAT.KI
7               4
1
2
2
4
4
6
```



3. feladat: Zárnyitogató (12 pont)

Egy zár három körlemez tárcsából áll (A, B és C), amelyek közös tengely körül forgathatók. Az egyes tárcsákon körben az 1,..,N egész számok vannak felírva. A tárcsák külön-külön és együtt is forgathatók mindkét irányban. Tehát a zárat háromféleképpen fordíthatjuk el: egyszerre egy tárcsát (A, B vagy C), bármely kettő tárcsát egyszerre (egy irányban) (AB, vagy AC vagy BC), illetve a három tárcsát egyszerre (ABC) forgatva egy irányba.

Egy lépésnek a fenti forgatások valamelyikét nevezzük, tetszőleges irányba egy egység elfordulással.

Készíts programot, amely a zár egy adott kiinduló helyzetéből ([a;b;c]) megadja a legrövidebb forgatási sorozat hosszát, amivel az [1;1;1] pozícióba juthatunk!

A ZAR.BE állomány első sorában van az N (1≤N≤100) érték. A második sorban három szám van, egy-egy szóközzel elválasztva, a három tárcsán pillanatnyilag beállított szám.

A ZAR.KI állományba a legrövidebb lépéssorozat hosszát kell írni, amellyel a zár kinyitható, azaz az [1;1;1] pozícióba hozható!

Példa:

```
ZAR.BE          ZAR.KI
6               4
1 5 3
```

Például egy lehetséges 4 hosszúságú lépéssorozat: BC-t kétszer forgatva csökkenő irányba, majd B-t kétszer forgatva csökkenő irányba.

4. feladat: Lift (21 pont)

Egy sokemeletes házban szokatlan módon üzemeltetik a liftet. A lift az első szintről indult és mindig felmegy a legfelső szintre, majd visszatér az első szintre. Menet közben megáll minden olyan szinten, amelyik úticélja valamelyik liftben tartózkodó utasnak. Hasonlóan, olyan szinten is megáll,

ahonnan utazni szándékozik valaki az aktuális irányban, feltéve, hogy még befér a liftbe (figyelembe véve az adott szinten kiszállókat).

Készíts programot, amely kiszámítja, hogy legkevesebb hány menet (1 menet = egyszer fel megy, majd lejön) szükséges ahhoz, hogy minden várakozó embert elszállítson a lift!

A LIFT.BE állomány első sorában két szám van: az épület szintjeinek száma (2≤N≤100), és a lift kapacitása (1≤K≤10). A további N sor tartalmazza az egyes szinteken várakozó emberek adatait. Az állomány i-edik sorában azoknak a szinteknek a sorszáma van felsorolva, ahová az i-1–edik szintről utazni akarnak. A felsorolást minden sorban egy 0 szám zárja. Minden sorban legfeljebb 200 szám lehet, tetszőleges sorrendben.

A LIFT.KI állomány egyetlen sort tartalmazzon, a legkevesebb menetek számát, amely az emberek elszállításához szükséges!

Példa:

LIFT.BE	LIFT.KI
6 2 2 3 2 0 1 3 0 1 2 0 2 5 0 3 6 2 0 1 2 3 0	3

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Lift (15 pont)

Egy sokemeletes házban szokatlan módon üzemeltetik a liftet. A lift az első szintről indul és mindig felmegy a legfelső szintre, majd visszatér az első szintre. Menet közben megáll minden olyan szinten, amelyik uticélja valamelyik liftben tartózkodó utasnak. Hasonlóan, olyan szinten is megáll, ahonnan utazni szándékozik valaki az aktuális irányban, feltéve, hogy még befér a liftbe (figyelembe véve az adott szinten kiszállókat).

Készíts programot, amely kiszámítja, hogy legkevesebb hány menet (1 menet = egyszer felmegy, majd lejön) szükséges ahhoz, hogy minden várakozó embert elszállítson a lift!

A LIFT.BE állomány első sorában két szám van: az épület szintjeinek száma (2≤N≤500), és a lift kapacitása(1≤K≤20). A további N sor tartalmazza az egyes szinteken várakozó emberek adatait. Az állomány i-edik sorában azoknak a szinteknek a sorszáma van felsorolva, ahová az i-1–edik szintről utazni akarnak. Minden sorban legfeljebb 500 szám lehet, tetszőleges sorrendben, a felsorolást egy 0 szám zárja.

A LIFT.KI állomány egyetlen sort tartalmazzon, a legkevesebb menetek számát, amely az emberek elszállításához szükséges!

Példa:

```
LIFT.BE              LIFT.KI
6 2                  3
2 3 2 0
1 3 0
1 2 0
2 5 0
3 6 2 0
1 2 3 0
```

2. feladat: Kockavilág (15 pont)

Van N db azonos méretű kockánk (1, 2, ..., N számokkal jelöljük). A kockák vagy az asztalon vannak, vagy egy másik kocka tetején.

Van egy robotkar, ami képes fentről megfogni a legfelső kockát és azt egy másik kocka tetejére vagy az asztalra helyezni. A cél az, hogy a robotkar segítségével úgy mozgassuk a kockákat, hogy a 1. legyen legalul, a 2. az 1. kockán legyen, stb.

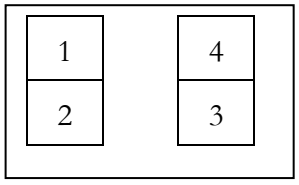
A KOCKA.BE állomány első sorában a kockák száma ($1 \le N \le 100$) van. A további N sorban található, hogy az adott kocka melyik másik kocka tetején van. Ez a szám 0, ha a kocka az asztalon van. Tehát az állomány i-edik sorában lévő szám azt adja meg, hogy az i-1–edik kocka melyik kocka tetején van. Ha egy kocka a másik tetején van, az csak úgy lehet, hogy az érintkező oldaluk teljesen fedi egymást.

A KOCKA.KI állomány első sorába a HIBAS szöveget kell írni, ha a KOCKA.BE valami oknál fogva nem megfelelő, egyébként a HELYES szót!

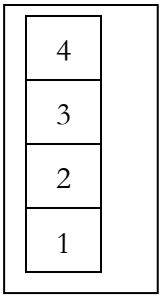
Az állomány második sorától kezdődően a legkevesebb lépésszámú megoldást kell írni! (Ha több ilyen is van, akkor csak az egyiket.) Minden sorban két szám szerepeljen: melyik kockát melyikre kell tenni!

Példa:

```
KOCKA.BE
4
2
0
0
3
```




```
KOCKA.KI
HELYES
1 0
2 1
4 0
3 2
4 3
```



3. feladat: Malacpersely(15 pont)

Mohó Marci malacperselyben gyűjti pénzét. Csak fémpénzeket rakott a perselybe, de nem jegyezte fel, hogy milyeneket. Felírta azonban az üres persely súlyát, így meg tudja állapítani a perselyben lévő pénzek összsúlyát. Ismeri továbbá az egyes pénzérmék egyedi súlyát és értékét. Szeretné kiszámítani, hogy mennyi az a legkisebb érték, amelyet a perselye biztosan tartalmaz. Egy adott típusú pénzérméből (címletből) több is lehet a perselyben.

Készíts programot, amely kiszámítja, hogy legkevesebb mekkora értéket tartalmaz a malacpersely!

A MALAC.BE állomány első sorában a perselyben lévő pénzek összsúlya van ($1 \le S \le 10\,000$). A második sorban a pénzérme fajták (címletek) száma ($1 \le N \le 100$) található. A további N sor mindegyike két pozitív egész számot tartalmaz: az első szám egy pénzérme értéke (nem nagyobb, mint 200), a második szám pedig a pénzérme súlya (nem nagyobb, mint 1000).

A MALAC.KI állomány egyetlen sort tartalmazzon, azt a legkisebb értéket, amelyet a malacpersely biztosan tartalmaz!

Példa:

```
MALAC.BE              MALAC.KI
15                    8     (6 db 1 forintos és 1 db 2 forintos)
4
1 2
2 3
5 6
10 4
```

4. feladat: Raktár (15 pont)

Egy raktár alapterületét négyzetrácsokra osztották be. Minden mező vagy polcokat tartalmaz, vagy üres. Közlekedni természetesen csak az üres mezőkön lehet, átlósan lépni nem lehet. A beosztást úgy alakították ki, hogy bármely két üres mező között pontosan egy út van.

Készíts programot, amely kiszámítja a raktárban a lehetséges leghosszabb útvonal hosszát.

A RAKTAR.BE bemeneti állomány első sorában két szám van ($2 \leq M,N \leq 200$). M a négyzetrácsban az oszlopok száma, N pedig a sorok száma. A további N sor mindegyike pontosan M karaktert tartalmaz (a karakterek között nincs szóköz).A # karakter foglalt, a . (pont) karakter pedig szabad mezőt jelöl.

A RAKTAR.KI állomány egyetlen sort tartalmazzon, a lehetséges leghosszabb útvonal hosszát!

Egy útvonal hossza az útvonalban lévő mezők száma (a két végpontot is beleértve) mínusz 1.

Példa:

```
RAKTAR.BE             RAKTAR.KI
6 5                   12
..#.#.
#.....
..##.#
.#....
.#.#.#
```

5. feladat: Üzenetek (15 pont)

A Kozmosz Rt. a stratégiai fontosságú üzenetek továbbítására saját rendszert dolgozott ki. Ha valaki, aki részt vesz a rendszerben és üzenetet kap, köteles azt továbbítani a számára előírt embereknek. A társaságnál az eziránytú kötelezettséget a következőképpen jelölték:

```
János(Géza,István,Mónika)
Géza(Éva,Lajos)
```

Jelentésük: Jánosnak továbbítania kell az üzenetet Gézának, Istvánnak és Mónikának, illetve Gézának továbbítania kell az üzenetet Évának és Lajosnak.

A társaság ezen szabályokat egy globális, összevont üzenetközvetítési szabályzattal írja le, ami a példa esetében:

```
János(Géza(Éva,Lajos),István,Mónika).
```

Az első ember (János) fogja az üzenetet megkapni a vezérigazgatóságtól, majd továbbítja azokat a számára kiírt embereknek, akik szintén továbbadják azt. Azon emberek, akiknek nincs kijelölve

senki, természetesen nem adják tovább az üzenetet senkinek. A társaság az üzenetközvetítés hatékonyságáról szeretne informálódni, megadott lánc esetén.

Készíts programot, amely kiszámolja az alábbiakat:

A. Egy embernek maximum hány másiknak kell közvetlenül átadnia az üzenetet?

B. Legfeljebb hány emberen keresztül jut el az üzenet valakihez?

C. Hány olyan ember van, akinek nem kell továbbítania az üzenetet?

Az UZENET.BE állomány első sora tartalmazza a szabályzatot, melyben maximum 1000 ember neve szerepel. A neveket az angol abc kis és nagy betűi jelölik, a szabályzat nem tartalmaz szóköz karaktert. A szabályzatban legalább egy ember szerepel. A szabályzatot leíró karaktersorozatot a # karakter zárja.

Az UZENET.KI állomány első sorába az A, a második sorába a B, a harmadik sorába pedig a C kérdésre adott választ kell írni!

Példa:

```
UZENET.BE:
Miklos(Peter(Balazs,Zsoka),Eva,Ferenc(Agi(Teri,Zsu-
zsa),Laci,Magdi,Dora))#
UZENET.KI:
4           (mert Ferenc 4 embernek adja)
3           (mert Terihez Mikloson, Ferencen és Agin keresztül jut el)
8
```

2000. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Villamos (25 pont)

A Villamos-közlekedési Vállalat (VKV) felmérést végzett a villamosok kihasználásáról, melyet számítógéppel kell feldolgozni. A villamos-vonalon N állomás van, beleértve az induló- és a végállomást is. Egy út során a villamosvezetőnek meg kellett számolnia minden állomáson a fel- és a leszállókat, s neked ezekből az adatokból kell adott jellemzőket kiszámolnod.

Készíts programot, amely beolvassa az állomások számát ($2 \le N \le 100$), a vezető által adott számokat (N*2 adat, mindegyik pozitív), majd belőlük a következőket határozza meg és írja ki a képernyőre:

- Tartalmilag helyesek-e a vezető által adott számok? (Formailag helyesek – azaz mindegyik pozitív – de például az induló állomáson a leszállók száma csak 0 lehet, minden más tartalmilag hibás.) Ha hibásak, akkor a program írja ki, hogy hibásak az adatok!
- Ha helyesek az adatok, akkor adja meg az alábbiakat:
  A. Hány ember utazott összesen a villamoson?
  B. Mely állomásokon szállt le a villamosról az összes utas?
  C. Mi volt a villamoson a maximális utasszám?
  D. Hány állomásközi szakaszt tett meg a villamos úgy, hogy egyetlen utas sem volt rajta?

Példa:

```
Bemenet: 5 állomás
    Felszállók: 5 3 0 2 0
    Leszállók:  0 4 4 0 2
A: Összesen 10 ember utazott a villamoson.
B: Mindenki leszállt a 3. és az 5. állomáson.
C: A maximális utasszám 5 volt.
D: 1 szakaszon nem volt utas.
```

Megjegyzés: a 3. és a 4. állomás között senki sem volt a villamoson.

2. feladat: Mondat-csavaró (20 pont)

Készíts programot, ami egy beolvasott mondattal tud többféle műveletet végezni! A mondat kisbetűkből és szóközökből áll, a szavakat pontosan egy szóköz választja el egymástól, az első szó előtt, valamint az utolsó után nincs szóköz.

A programod az alábbi átalakításokat tudja elvégezni:

A. A mondatot úgy írja ki egy sorba, hogy a szavai sorrendjét megfordítja.

B. A mondatot úgy írja ki egy sorba, hogy csak a páratlan sorszámú szavai szerepelhetnek benne.

C. Csak minden szó kezdőbetűjét írja ki, de azt nagybetűvel, szóközökkel elválasztva.

D. A mondat szavait az eredeti sorrendben írja ki, de a szavak betűsorrendjét megfordítja.

Példa:

```
Bemenet: az ibafai papnak fapipája van
A. van fapipája papnak ibafai az
B. az papnak van
C. A I P F V
D. za iafabi kanpap ajápipaf nav
```

3. feladat: Húsvét (30 pont)

Készíts programot, amely megadja, hogy egy évben milyen napra esik húsvét, illetve pünkösd!

A húsvét meghatározásának szabálya:

> *Húsvét a tavaszi napéjegyenlőség (március 21.) utáni első holdtölte utáni első vasárnap, illetve hétfő. A holdtöltéktől egymástól 29 és fél napra vannak.*

Megjegyzés: Ha március 21. holdtölte is és vasárnap is, akkor már egyben húsvétvasárnap is.

Például 1991. január 1. kedd volt, az első holdtölte: január 30.-án délelőtt volt, akkor a következők: február 28. délután, március 30. délelőtt.

A pünkösd meghatározásának szabálya:

> *Pünkösdvasárnap a húsvétvasárnap utáni hetedik vasárnap, pünkösdhétfő pedig az azt követő hétfő.*

A program olvassa be, hogy melyik évben vagyunk, az év első napja a hét mely napjára esik, illetve, hogy január hányadikán van az első holdtölte, s aznap délelőtt vagy délután, s ezek alapján írja ki húsvétvasárnap és húsvéthétfő, valamint pünkösdvasárnap és pünkösdhétfő dátumát!

Példa:

```
Bemenet:  Év: 1991  Első nap: kedd
          Holdtölte: 30. délelőtt
```

```
Eredmény: Húsvétvasárnap: Március 31.
          Húsvéthétfő: Április 1.
          Pünkösdvasárnap: Május 19.
          Pünkösdhétfő: Május 20.
```

Kilencedik-tizedik osztályosok

1. feladat: Autópálya (23 pont)

Bergengóciában nevezetességeit egy kör alakú autópálya köti össze, amely mentén benzinkutak sorakoznak. Berg Egon elhatározta, hogy körbeautózik az úton. Az autóját üres tankkal tudja csak szállítani az egyik benzinkúthoz. Tudjuk, hogy melyik benzinkútnál mennyi benzin van és ismerjük a benzinkutak egymástól való távolságát.

Készíts programot, amely megadja, hogy melyik kúttól kell indulnia Egonnak az ott található teljes benzinkészlettel (biztosan magával tudja vinni), hogy úgy tudjon körbeautózni, hogy autójából ne fogyjon ki a benzin. Az autópálya egyirányú, az I. benzinkúttól csak az I+1. felé lehet indulni, valamint az N.-től az 1. felé.

A KORUT.BE állomány első sorában a kutak száma ($1 \le N \le 16\,000$) van, és hogy az autó egy liter benzinnel hány kilométert tud megtenni ($1 \le M \le 100$). A következő N sor mindegyike két egész számot tartalmaz; az első a következő kút távolságát adja meg kilométerben (legfeljebb 1000 km), a második pedig az itt fellelhető benzin mennyiségét (legfeljebb 200 liter).

A KORUT.KI állomány első sorába az IGEN szót kell írni, ha valamelyik kúttól kezdve az autópálya körbeutazható, egyébként pedig a NEM szót! A második sorba azon kutak számát kell írni, melyek bármelyikéből indulva körbe lehet autózni! Ha sehonnan sem lehet körbejutni, akkor ide annak a kútnak a sorszámát kell írni, ahonnan a legmesszebbre el lehet jutni, azaz a legtöbb kutat lehet érinteni! (Ha több megoldás is van, közülük egyet kell megadni!)

Példa:

KORUT.BE	KORUT.KI
8 10	IGEN
200 25	4 6 5
200 15	
100 5	
200 20	
300 30	
400 45	
200 20	
200 20	

2. feladat: Vállalatok (30 pont)

Egy vállalat sorsát szeretnénk követni az alábbiakban. Kezdetben egyetlen vállalat létezik. Tetszőleges időpontokban a vállalatot szétbonthatják több önálló vállalatra, illetve önállóan létező vállalatokat újra összevonhatnak. A keletkező vállalatok neve nem lehet azonos egyetlen, korábban létező vállalatéval sem. Egyesüléskor az egyesült, szétbontáskor pedig az egyik keletkező vállalat megtarthatja egyik közvetlen elődje nevét. Ha szétbomlásnál egyetlen vállalat sem keletkezik, akkor a vállalat utód nélkül szűnt meg. Ha egyesülésnél nincs egyesülő, akkor pedig új vállalat keletkezett. Egy vállalattal egy napon kétféle változás nem lehetséges. Kezdetben egyetlen vállalat létezik.

Készíts programot, amely meghatározza, hogy egy ember, aki végig ugyanazon a helyen dolgozik, maximum hány vállalat alkalmazottja lehetett az idők során, valamint hogy mikor volt az, amikor a kiinduló vállalat a legtöbb részre oszlott, s ekkor hány önálló vállalat volt!

A VALLALAT.BE állomány első sorában az egyesülések/szétbomlások (1≤N≤500) száma, valamint a kezdetben létező vállalat neve található. A következő N sorban vannak a változások. Mindegyik ilyen sor első karaktere plusz-jel (+), ha vállalat-egyesülést ír le, illetve mínusz-jel (-), ha vállalat szétbontást. A jelet annak a vállalatnak a neve követi, ami az egyesült vállalat neve (maximum 10 karakter) lesz, illetve ami a szétbomló vállalat neve volt. Ettől szóközzel elválasztva annak a napnak a sorszáma szerepel, amikor a változás történt. Ettől szóközökkel elválasztva következik a változásban résztvevő vállalatok neve (amik egyesülnek, illetve keletkeznek). A változások időben növekvő sorrendben szerepelnek az állományban. Legfeljebb 2000 különböző vállalat neve szerepel az állományban.

A VALLALAT.KI állomány első sorában azon vállalatok maximális számát kell írni, ahol egy ember (munkahelyváltás nélkül) dolgozhatott! A második sorba szóközökkel elválasztva az egyes vállalatok neve kerüljön, ahol dolgozhatott! (Ha több megoldás is van, akkor csak egyet kell megadni!) A harmadik sorba azt a napsorszámot kell írni, amikor a legtöbb vállalat létezett, s tőle egy szóközzel elválasztva az ekkori vállalatszámot! (Ha több megoldás is van, akkor csak egyet kell megadni!)

Példa:

```
VALLALAT.BE          VALLALAT.KI
4 ELSO               5
-ELSO 4 a b c        utolso e d b ELSO
+d 6 b c             8 4
-d 8 e f g
+utolso 10 e f
```

3. feladat: Alkimisták (22 pont)

Az alkimisták hosszas kísérleteiket arról, hogy milyen anyagok milyenekké alakíthatók át, gondosan feljegyezték könyveikbe. Bejegyzéseik a lehető legegyszerűbbek voltak: Megadták a kiindulási anyag nevét, majd azt az összetevőt (ún. katalizátort), amit hozzákeverve ehhez létrejött valamilyen végtermék. A bejegyzéseket tanulmányozva megállapították, hogy egyetlen anyag sem állítható elő önmagából, egy vagy több lépésben sem, továbbá nincs két különböző bejegyzés, amelyekben mind az első, mind a második anyag megegyezne. Az anyagok neveit számokkal, míg a katalizátorokat az angol ABC nagy betűivel jelölték A-tól Z-ig.

Egy ilyen bejegyzés-sorozatra példa:

1 A 2 (1 anyagból 2 és

1 A 3 3 keletkezik, ha A-t keverünk hozzá)

1 B 4 (Ha 1-hez B-t adagolunk, akkor 4 keletkezik)

3 F 0 (Ha a létrejött 3-hoz F-et keverünk akkor létrejön 0)

Az alkimisták aranyat szerettek volna előállítani vasból. Feljegyzéseikben az aranyat 0-val, míg a vasat 1-el jelölték.

Készíts programot, amely megadja, hogy egy adott bejegyzés-sorozatban melyek azok a katalizátorok, amelyek feltétlenül szükségesek ahhoz, hogy az alkimista szerint aranyat vasból elő lehessen állítani.

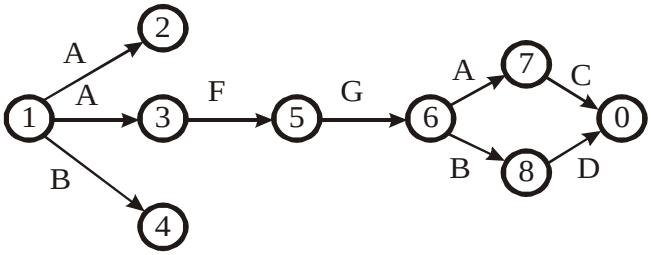
Az ARANY.BE állomány első sorában megadjuk, hogy hány bejegyzés szerepel a sorozatban, a többi sora pedig a fenti példában szereplő formátumban tartalmazza az egyes bejegyzéseket (a kiindulási anyag száma, szóköz, a katalizátor betűjele, szóköz és a végtermék száma). A használt anyagok száma maximum 200, a katalizátoroké pedig 26. A bejegyzések száma legfeljebb 1000.

Az ARANY.KI állományba a nélkülözhetetlen katalizátorok betűjelét kell írni, egy sorban szóközzel elválasztva! Ha aranyat nem lehet előállítani semmilyen katalizátorral, akkor NEM LEHET szerepeljen a kimenetben! Ha egyik katalizátor sem nélkülözhetetlen, akkor EGYIK SEM KELL legyen a sor tartalma!

Példa:

```
ARANY.BE
9
1 A 2
1 A 3
1 B 4
3 F 5
5 G 6
6 A 7
6 B 8
7 C 0
8 D 0
```

```
ARANY.KI
A F G
```



Tizenegyedik-tizenharmadik osztályosok

1. feladat: Térkép (30 pont)

Egy térképet egy NxM-es mátrixban ábrázolunk. A városokat a mátrixban a 2-es számjegy, az utakat pedig az 1-es számjegy jelzi. A többi ponthoz tartozó érték 0. Az utak minden pontból a 4 szomszédos pont irányában folytatódhatnak, azaz átlósan lépni nem lehet. Egy város több 2-es értékű pontból is állhat, de két város sehol sem érintkezhet egymással, azaz nincs szomszédos pontjuk.

Készíts programot, amely két város esetén megadja a városok területét (hány 2-es értékű pontból áll), valamint két a közöttük vezető legrövidebb út hosszát az alábbi háromféle módon:

- a legrövidebb út az az út, ami a legkevesebb várost érint a kiindulásin kívül; a hossza pedig az érintett városok szám
- a legrövidebb út az az út, ami legkevesebb, egyik városhoz sem tartozó (1-es értékű) közbülső ponton halad át, a hossza pedig ezen pontok száma;
- a legrövidebb út az az út, amin a leggyorsabban el lehet érni a másik városba, feltételezve, hogy városban feleakkora a sebesség, mint a városok közötti utakon, azaz az 1-essel jelölt pontot 1, a 2-essel jelölt pontot pedig 2 időegység alatt lehet elhagyni; s a hossza az út megtételéhez szükséges idő.

A TERKEP.BE állomány első sorában sorok és oszlopok száma van ($1 \le N,M \le 100$). A további N sor mindegyike pontosan M számjegyet tartalmaz (csak 0, 1 vagy 2 lehet) szóközök nélkül, a térkép egyes sorai leírását. Az állomány utolsó sora két város indexeit ((X,Y) és (U,V)), azaz négy számot tartalmaz ($1 \le X,U \le N$, $1 \le Y,V \le M$), s a feladat az (X,Y)-ból (U,V)-be vezető legrövidebb út megtalálása. Az első sor első (bal oldali) elemének koordinátái (1,1), az N. sor M. elemének pedig (N, M).

A TERKEP.KI állományba négy sort kell írni! Az első sorban az (X,Y), illetve az (U,V) pontot tartalmazó város területét kell írni! A következő három sorban egyetlen szám szerepeljen: a feladatban szereplő három megfogalmazásbeli legrövidebb út hossza! Ha nincs út a két város között, akkor -1-et kell írni mindhárom sorba!

Példa:

```
TERKEP.BE           TERKEP.KI
9 10                3 5
2200000000          2
0200000000          8
0110000000          22
0010000000
0011200000
0001220000
0000211122
0000000122
0000000112
1 1 9 10
```

2. feladat: Karaván (30 pont)

Egy sivatagban N város található, melyek között az év egy időszakában naponta indulnak karavánok. Ezen az időszakon kívül azonban egyetlen sem indul.

Készíts programot, amely meghatározza két adott, A B városra és H határidőre a következőket:

- Legkorábban mikorra érhetünk az A városból a B városba;
- Legalább hány nap kell ahhoz, hogy az A városból a B városba érjünk, ha nincs megkötés sem az indulási, sem az érkezési időre;
- Legkésőbb melyik napon kell elindulni az A városból, hogy legkésőbb a megadott H határidőig a B városba érjünk.

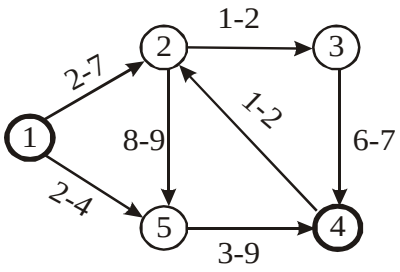
A KARAVAN.BE állomány első sorában a városok száma (1≤N≤100) és a karaván-kapcsolatok száma (1≤M≤5000) van. A következő M sor mindegyikében két város-sorszám (X, Y) és két napsorszám (P,Q) van egy-egy szóközzel elválasztva, ami azt jelenti, hogy az X. városból az Y városba az év P. és Q. napja között (P-t és Q-t beleértve) indulnak karavánok. Az állomány utolsó sorában az indulási (A) és az érkezési (B) hely sorszáma van, valamint annak a napnak az éven belüli H sorszáma, amikor legkésőbb meg kell érkezni a B. városba, egy-egy szóközzel elválasztva. A karavánok mindig reggel indulnak és még aznap este megérkeznek a célállomásra.

A KARAVAN.KI állományba három sort kell írni! Minden sorban egyetlen szám szerepeljen: a részfeladat megoldásának értéke! Ha egy részfeladatnak nem létezik megoldása, akkor a –1 számot kell kiírni!

Példa:

```
KARAVAN.BE      KARAVAN.KI
5 7             3       (2. nap: 1→5, 3. nap: 5→4)
1 2 2 7         2
1 5 2 4         7       (7. nap: 1→2, 8. nap: 2→5, 9. nap: 5→4)
2 3 1 2
2 5 8 9
3 4 6 7
5 4 3 9
4 2 1 2
1 4 10
```

3. feladat: Dominó (15 pont)



Dominóval sokféle játékot lehet játszani. Mohó Marci kedvenc dominós játéka a következő. Először véletlenszerűen sorba rakja a felhasználható dominókat. A játék célja az, hogy a lehető leghosszabb illeszkedő sorozatot képezzen a felhasználható dominókból. A játékszabály szerint minden lépésben csak a felhasználható dominósor első (bal oldali) elemét veheti és vagy elveti (félrerakja, de később nem veheti), vagy a már képzett illeszkedő sorozat bal vagy jobb végéhez teszi, feltéve, hogy az adott oldalával illeszkedik (megegyezik a pöttyök száma a két dominó érintkező oldalán). Az aktuális dominót mindkét oldalával próbálhatja illeszteni. A játék úgy kezdődik, hogy az első dominót ki kell raknia.

Készíts programot, amely meghatározza a kirakható leghosszabb illeszkedő dominósor hosszát!

A DOMINO.BE állomány első sorában a felhasználható dominók száma ($1 \le N \le 100\,000$) van. A következő N sor mindegyikében egy dominó leírása, azaz két szám, X Y ($0 \le X,Y \le 9$) van. Bármely dominó (számpár) többször is szerepelhet az állományban, és az állomány nem feltétlenül tartalmaz minden lehetséges dominót.

A DOMINO.KI állományba egyetlen számot kell írni, a kirakható leghosszabb illeszkedő dominósor hosszát!

Példa:

DOMINO.BE	DOMINO.KI
6 1 2 1 6 2 3 1 4 2 3 4 3	5

A verseny végeredménye:

I. korcsoport

1. Ritzinger Péter	Apor Vilmos Iskolaközpont, Gyõr
Sztupák Szilárd	Herman Ottó Gimnázium, Miskolc
3. Bauer Péter	Bolyai Gyakorló Általános Iskola és Gimnázium, Szombathely
4. Rácz Béla András	Fazekas Mihály Gimnázium, Budapest
5. Gruber László	KLTE Arany János Gyakorló Általános Iskola, Debrecen
6. Fehér Gábor	Berzsenyi Dániel Gimnázium, Budapest
Paulin Roland	Fazekas Mihály Gimnázium, Budapest
8. Juhász Máté	Fazekas Mihály Gimnázium, Budapest
Rák Ádám	Fényi Gyula Jezsuita Gimnázium, Miskolc
Ujhelyi Zoltán	Petõfi Sándor Általános Iskola, Vác

II. korcsoport

1. Pallos Péter	Fazekas Mihály Gimnázium, Budapest
2. Pszota Zsolt	Boronkay György Szakközépiskola, Vác
3. Zavarkó Gábor	Földes Ferenc Gimnázium, Miskolc
4. Hargitai Gábor	Bolyai János Gimnázium, Ócsa
5. Szabó András	Kazinczy Ferenc Gimnázium, Gyõr
6. Siroki László	Fazekas Mihály Gimnázium, Debrecen
7. Szebenyi Zoltán	Ságvári Endre Gimnázium, Szeged
8. Szeredi Dániel	Veres Péter Gimnázium, Budapest
9. Marton József	Ságvári Endre Gimnázium, Szeged
Balogh János	Táncsics Mihály Gimnázium, Kaposvár
Barta Gábor	Neumann János Szakközépiskola és Gimnázium, Eger

III. korcsoport

1. Rokob András	Földes Ferenc Gimnázium, Miskolc
2. Dezsõ Balázs	Teleki Blanka Gimnázium, Székesfehérvár
3. Gyebnár Gábor	Ságvári Endre Gimnázium, Szeged
4. Novák Ádám	Neumann János Szakközépiskola és Gimnázium, Eger
5. Soós István	Kanizsai Dorottya Gimnázium, Szombathely
6. Ritter Ádám	Fazekas Mihály Gimnázium, Budapest
7. Csillag Kristóf	Karacs Ferenc Gimnázium, Püspökladány
8. Flach Attila	Ságvári Endre Gimnázium, Szeged
9. Sáfár Szilveszter	Ságvári Endre Gimnázium, Szeged
10. Pataki Gergely	Neumann János Szakközépiskola és Gimnázium, Eger
Máthé András	Apáczai Csere János Gimnázium, Budapest

2001. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mókusok (28 pont)

Mókusok télire diót és mogyorót gyűjtenek az erdőben (N darabot). Amit találnak, azt lyukakba rejtik, egy lyukba egyet. A főmókus szeretné ésszerűen elrendezni a gyűjteményt, s ezért az alábbi algoritmus alapján rendezi át a diók és mogyorók sorrendjét:

```
Eljárás:
  i:=1; j:=N
  Tedd zsebre az 1. lyukban levő valamit!
  Ciklus amíg i<j
    Ciklus amíg i<j és a j. lyukban mogyoró van
      j:=j-1
    Ciklus vége
    Ha i<j akkor Tedd át a j. lyukból az i. lyukba!
    Ciklus amíg i<j és az i. lyukban dió van
      i:=i+1
    Ciklus vége
    Ha i<j akkor Tedd át az i. lyukból a j. lyukba!
  Ciklus vége
  Tedd a zsebedből az i. lyukba!
Eljárás vége.
```

A. Milyen sorrendben lesznek az algoritmus végén a diók és mogyorók?

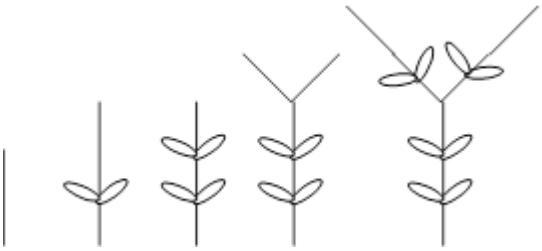
B. Az algoritmus alapján hogyan lehetne megmondani, hogy hány diót, illetve hány mogyorót gyűjtöttek?

C. Az algoritmus milyen elhelyezkedésű diókat, illetve mogyorókat hagy a helyén?

D. Add meg, hogy legjobb, illetve legrosszabb esetben hány gyümölcsöt (diót és mogyorót) kell mozgatni és milyen ekkor a kezdeti elrendezés?

2. feladat: Szobanövény (17 pont)

Egy szobanövény az elültetése utáni évben két zöld levelet növeszt. A második évben 2 piros levelet, majd a harmadik évben elágazik kétfelé. A negyedik évtől kezdve az új ágakon ugyanaz történik, mint az eredeti növényen, azaz megjelenik két zöld levél, egy évre rá két piros levél, majd újabb 1 év múlva ezek az ágak is szétágaznak kétfelé. (Mint az ábrán látszik, az 1 éves növénynek 2 levele van, a 2 és a 3 évesnek 4, …)



A. Hány piros levele lesz a 6 éves növénynek? (A 3 évesnek 2 van.)

B. Hány zöld levele lesz a 7 éves növénynek? (A 3 évesnek 2 van.)

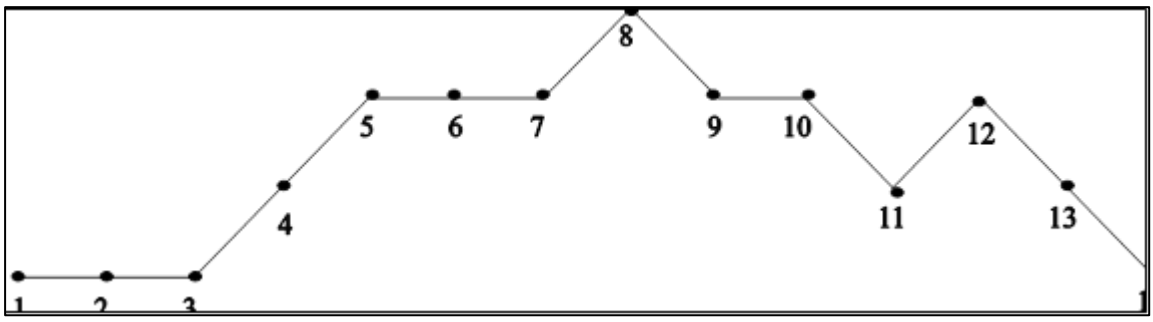
C. Hány ágvége lesz összesen a 8 éves növénynek? (A 3 évesnek 2 van.)

D. Hányadik évben lépi túl a levelek száma a 200-at?

E. Milyen években lesz a növénynek több zöld levele, mint piros?

3. feladat: Hegymászó (24 pont)

Egy hegymászó az útja során N pontban feljegyezte, hogy milyen tengerszint feletti magasságban járt. Ezt mutatja az alábbi ábra.



Az i-edik pontban mért tengerszint feletti magasságot X_i-vel jelöljük. Az egyes pontok különböző nehézségű szakaszok határán vannak (pl. emelkedő, sík, …). Az alábbi algoritmus ezeket sorolja be 3 csoportba:

```
Pontok:
  Ciklus i=2-től N-1-ig
    Ha (Xi-Xi-1)*(Xi+1-Xi)>0 akkor Ki: i,' 1. típusú' {*}
    Ha (Xi-Xi-1)*(Xi+1-Xi)=0 akkor Ki: i,' 2. típusú' {**}
    Ha (Xi-Xi-1)*(Xi+1-Xi)<0 akkor Ki: i,' 3. típusú' {***}
  Ciklus vége
Eljárás vége.
```

A. Az ábrán látható pontok közül milyen sorszámúakat sorol be az algoritmus az 1, 2, illetve 3 típusú pontok közé?

A *-gal jelölt kiírások helyére újabb algoritmusrészleteket teszünk a pontok további osztályozása érdekében:

```
{*}: Ha (Xi-Xi-1) >0 akkor Ki: i,' 1/A típusú'
                     különben Ki: i,' 1/B típusú'

{**}:   Elágazás
          (Xi-Xi-1)>0 esetén Ki: i,' 2/A típusú'
          (Xi+1-Xi)>0 esetén Ki: i,' 2/B típusú'
          (Xi-1-Xi)>0 esetén Ki: i,' 2/C típusú'
          (Xi-Xi+1)>0 esetén Ki: i,' 2/D típusú'
         egyéb esetben Ki: i,' 2/E típusú'
        Elágazás vége

{***}:  Ha (Xi-Xi-1) >0 akkor Ki: i,' 3/A típusú'
                       különben Ki: i,' 3/B típusú'
```

B. Az ábrán látható pontok közül milyen sorszámúakat sorol be az algoritmus az 1/A, 1/B, 2/A, …, 3/B típusú pontok közé?

4. feladat. Osztálybuli (31 pont)

Osztályod két bulit rendezett. A rendezvényre az osztálytársak különböző időpontokban érkeztek, illetve távoztak. Valaki pontosan feljegyezte minden résztvevő tanulóról, hogy az mikor érkezett, illetve távozott. Egy feljegyzés egy [a,b] számpár, ami azt jelenti, hogy a tanuló az a időponttól a b időpontig volt jelen a rendezvényen (beleértve az a és a b időpontokat is). Minden időpont a rendezvény kezdete óta eltelt idő másodpercben mérve.

Feljegyzés:

1. `[600, 1000],[1, 100], [100, 500], [500, 700], [400, 800], [650, 900], [300, 600], [66, 120]`

2. `[1, 1000], [4000, 6000], [500, 2113], [2345, 5000], [601, 1000], [5000, 6000], [2000, 3000], [2345, 3333], [3001, 4000], [3350, 5010], [4001, 5000], [6300, 9000], [5056, 5156], [5621, 6000], [3350, 5056], [6000 7000]`

A. Hányan voltak legtöbben egyszerre jelen a rendezvényen?

B. Adj meg egy olyan időpontot, amikor a legtöbben voltak egyszerre jelen.

C. Legalább hány fényképet kellett volna készíteni ahhoz, hogy mindenki szerepeljen legalább egy fényképen?

D. Adj meg a C. kérdésben szereplő számú időpontot, hogy ha ekkor készült volna egy-egy fénykép, akkor mindenki szerepelne legalább egy fényképen.

Kilencedik-tizedik osztályosok

1. feladat: Gráf (20 pont)

Egy irányított gráfot sorszámozott pontokkal és azokat összekötő élekkel adunk meg. A gráfot egy G mátrix írja le, amelyben G(i,j)=1, ha az i-edik pontból vezet él a j-edik pontba, s G(i,j)=0, ha nem. Az alábbi algoritmus egy N pontból álló gráf pontjainak tulajdonságait vizsgálja:

```
Algoritmus:
  i:=1; j:=N; k:=1
  Ciklus amíg i≠j
    Ha G(i,j)=1
        akkor  j:=j-1       {*}
      különben i:=i+1       {**}
  Ciklus vége
  Ciklus amíg k≤N és (G(i,k)=1 és G(k,i)=0 vagy i=k)
    k:=k+1
  Ciklus vége
  Ha k>N
       akkor s:=i       {***}
Eljárás vége.
```

A. Milyen tulajdonságú pontok esetén hajtódik végre a {*}-gal jelölt utasítás?

B. Milyen tulajdonságú pontok esetén hajtódik végre a {**}-gal jelölt utasítás?

C. Milyen tulajdonságú pontok esetén hajtódik végre a {***}-gal jelölt utasítás?

D. Lehetséges-e, hogy egy gráfban több olyan tulajdonságú pont is van, mint aminek a sorszámát a {***}-gal jelölt utasításban megadjuk? Magyarázd meg, miért!

2. feladat: Bankár-algoritmus (20 pont)

Egy kisvárosi bankban N ügyfél szeretne kölcsönt felvenni. Mindegyiknek ismerjük a hitelkeretét, amelyet előbb-utóbb teljesen kihasznál, de egyszerre csak annyit vesz fel, amennyire épp szüksége van. A bank nem akar annyi pénzt tartalékolni, amennyi a hitelkeretek összege, ugyanis azt feltételezi, hogy az egyes ügyfeleknek nem ugyanakkor van szükségük hitelre. Ha a bankban van elég pénz, akkor a bankár a hitelkérelmet elfogadhatja, ha nincs, akkor pedig a kérőt várakoztatja. A módszer problémája, ha minden ügyfél éppen újabb hitelt szeretne felvenni (ami még belefér a hitelkeretébe), de a bankban nincs pénz, így mindegyiket várakoztatni kellene. (Ekkor persze visszafizetni sem tudnak, így holtpont-helyzet alakul ki).

A bankár a következő stratégiát találta ki: az igényeket minden hitelkéréskor ellenőrzi, hogy a teljesítése esetén kerülhet-e holtpont-helyzetbe. Nem kerülhet abba, ha van legalább 1 valaki, akinek a maradék keretét oda tudja adni. Ha ilyen helyzetbe kerülne, akkor várakoztatja a hitelkérőt amíg ez a bizonytalan helyzet meg nem szűnik. Ha van valaki, aki több hitelt már nem kérhet, azaz előbb-utóbb visszafizeti a hitelt, akkor ez a helyzet megszűnhet. Ilyenkor a bankár a várakozók igényeit próbálja kielégíteni. Az igényeket az érkezésük sorrendjében igyekszik teljesíteni.

Az alábbi hitelkérsés-sor, hitelkeret és a bank kezdő tőkéje esetén add meg, hogy a hitelkéréseket és visszaadásokat milyen sorrendben teljesíti a bankár (a sorszámokat írd fel a megfelelő sorrendben)!

A hitelkeretek: Anna – 600, István – 500, Erika – 400, Ferenc – 700 dollár. A bankban kezdetben összesen 1000 dollár van.

A hitelkérések és visszafizetések ilyen sorrendben érkeznek:

```
1. Anna: 100
2. István: 100
3. Erika: 200
4. Ferenc: 400
5. István: 100
6. Erika: 200
7. Erika: -400 (visszafizet)
8. Anna: 100
9. Ferenc: 300
10. Ferenc: -700
11. Anna: 400
12. István: 300
13. Anna: -600
14. István: -500
```

3. feladat: Lemezhibák (12 pont)

Egy lemezegység egyetlen könyvtára N állományt, illetve M blokkot tartalmaz. Minden állomány valahány blokkban helyezkedik el, a blokkokat állományonként láncolva ábrázolják (külön a szabad blokkokat), minden blokknak pontosan 1 láncba kell tartoznia. Az állományszerkezet tartalmazhat láncolási hibákat. Az alábbi algoritmus ezeket próbálja felderíteni:

```
Hibafelderítés:
  Ciklus i=1-től M-ig
    T(i):=0; S(i):=0
  Ciklus vége
  Ciklus i=1-től N-ig
    j:=az i. állomány első blokkja
    Ciklus
      T(j):=T(j)+1; j:=az i. állomány következő blokkja
    amíg j valódi blokksorszám
    Ciklus vége
  Ciklus vége
  j:=első szabad blokk
  Ciklus
    S(j):=S(j)+1; j:=következő szabad blokk sorszáma
  amíg j valódi blokksorszám
  Ciklus vége
  Ciklus i=1-től M-ig
    Ha T(i)>0 és S(i)>0 akkor HIBA1
    Ha T(i)=0 és S(i)=0 akkor HIBA2
  Ciklus vége
Eljárás vége.
```

A. Milyen hibás esetben hajtódik végre a HIBA1 eljárás?

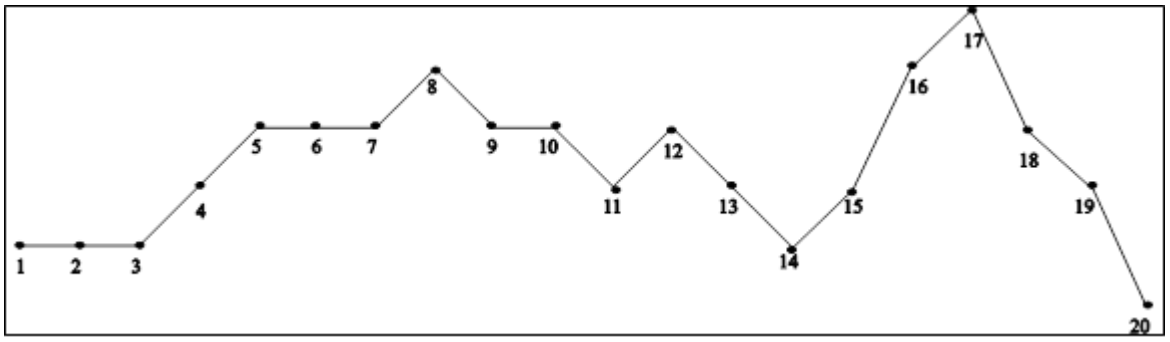
B. Milyen hibás esetben hajtódik végre a HIBA2 eljárás?

C. Mit kellene beszúrni a legutolsó ciklusba, ha figyelni szeretnénk, hogy egy blokk legfeljebb 1 állományhoz tartozzon?

D. Mit kellene beszúrni a legutolsó ciklusba, ha figyelni szeretnénk, hogy egy blokk csak egyszer szerepeljen a szabad blokkok között?

4. feladat: Hegymászó (28 pont)

Egy hegymászó az útja során N pontban feljegyezte, hogy milyen tengerszint feletti magasságban járt. Ezt mutatja az alábbi ábra.



Az i-edik pontban mért tengerszint feletti magasságot X_i-vel jelöljük. Az egyes pontok különböző nehézségű szakaszok határán vannak (pl. emelkedő, sík, …). Az alábbi algoritmus ezeket sorolja be 3 csoportba:

```
Pontok:
  Ciklus i=2-től N-1-ig
    Ha (Xi-Xi-1)*(Xi+1-Xi)>0 akkor Ki: i,' 1. típusú' {*}
    Ha (Xi-Xi-1)*(Xi+1-Xi)=0 akkor Ki: i,' 2. típusú' {**}
    Ha (Xi-Xi-1)*(Xi+1-Xi)<0 akkor Ki: i,' 3. típusú' {***}
  Ciklus vége
Eljárás vége.
```

A. Az ábrán látható pontok közül milyen sorszámúakat sorol be az algoritmus az 1, 2, illetve 3 típusú pontok közé?

A *-gal jelölt kiírások helyére újabb algoritmusrészleteket teszünk a pontok további osztályozása érdekében:

```
{*}:    Ha (Xi-Xi-1)>0 akkor
           Ha (Xi-Xi-1)+(Xi-Xi+1)>0 akkor Ki: i,' 1/A típusú'
           Ha (Xi-Xi-1)+(Xi-Xi+1)=0 akkor Ki: i,' 1/B típusú'
           Ha (Xi-Xi-1)+(Xi-Xi+1)<0 akkor Ki: i,' 1/C típusú'
        különben
           Ha (Xi-Xi-1)+(Xi-Xi+1)>0 akkor Ki: i,' 1/D típusú'
           Ha (Xi-Xi-1)+(Xi-Xi+1)=0 akkor Ki: i,' 1/E típusú'
           Ha (Xi-Xi-1)+(Xi-Xi+1)<0 akkor Ki: i,' 1/F típusú'
        Elágazás vége

{**}:   Elágazás
         (Xi-Xi-1)>0 esetén Ki: i,' 2/A típusú'
         (Xi+1-Xi)>0 esetén Ki: i,' 2/B típusú'
         (Xi-1-Xi)>0 esetén Ki: i,' 2/C típusú'
         (Xi-Xi+1)>0 esetén Ki: i,' 2/D típusú'
         egyéb esetben Ki: i,' 2/E típusú'
        Elágazás vége
```

```
{***}:  Ha (Xi-Xi-1) >0 akkor Ki: i,' 3/A típusú'
                       különben Ki: i,' 3/B típusú'
```

B. Az ábrán látható pontok közül milyen sorszámúakat sorol be az algoritmus az 1/A, 1/B, …, 3/B típusú pontok közé?

5. feladat: Kockarakás (20 pont)

A 7 törpe szabadidejében építőkockákból épít tornyokat. Különböző méretű építőkockákkal rendelkeznek, s a toronyépítésnek egyetlen szabálya van: kisebb kockára nagyobbat nem lehet tenni. Az építkezéshez N kockát használnak, az egyes kockát méretét K(i) jelöli (1≤K(i)≤99). A kockákat sorban helyezik el, s egy elhelyezett kockát már nem lehet mozgatni. A megoldásban M lesz a tornyok száma, T(j) pedig a j. torony tetején levő kocka mérete.

```
Hapci:
  M:=1; T(1):=K(1)
  Ciklus i=2-től N-ig
    j:=1
    Ciklus amíg j≤M és K(i)>T(j)
      j:=j+1
    Ciklus vége
    Ha j≤M akkor T(j):=K(i) különben M:=M+1; T(M):=K(i)
  Ciklus vége
Eljárás vége.
```

```
Tudor:
  M:=1; T(1):=K(1)
  Ciklus i=2-től N-ig
    mm:=T(1); jj:=1
    Ciklus j=2-től M-ig
      Ha T(j)>mm akkor mm:=T(j); jj:=j
    Ciklus vége
    Ha mm<K(i) akkor M:=M+1; T(M):=K(i) különben T(jj):=K(i)
  Ciklus vége
Eljárás vége.
```

```
Kuka:
  M:=1; T(1):=K(1)
  Ciklus i=2-től N-ig
    Ha T(1)<K(i) akkor M:=M+1; T(M):=K(i) különben T(1):=K(i)
  Ciklus vége
Eljárás vége.
```

A. Milyen elven választják ki a továbbépítendő tornyot az egyes törpék?

B. Állítsd sorrendbe a törpéket aszerint, hogy melyik tudja kevesebb toronyba elhelyezni a kockákat!

C. Milyen kockasorozatra építi mindhárom törpe ugyanazt az egy tornyot (N>2)?

D. Adj olyan kockasorozatot (N=5), amelyre a törpék mind különböző számú tornyokat építenek!

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Gráf (18 pont)

Egy irányított gráfot sorszámozott pontokkal és azokat összekötő élekkel adunk meg. A gráfot egy G mátrix írja le, amelyben G(i,j)=1, ha az i-edik pontból vezet él a j-edik pontba, s G(i,j)=0, ha nem. Az alábbi algoritmus egy N pontból álló gráf pontjainak tulajdonságait vizsgálja:

```
Algoritmus:
  i:=1; j:=N; k:=1
  Ciklus amíg i≠j
    Ha G(i,j)=1 akkor  j:=j-1       {*}
                különben i:=i+1     {**}
  Ciklus vége
  Ciklus amíg k≤N és (G(i,k)=1 és G(k,i)=0 vagy i=k)
    k:=k+1
  Ciklus vége
  Ha k>N akkor s:=i      {***}
Eljárás vége.
```

A. Milyen tulajdonságú pontok esetén hajtódik végre a {*}-gal jelölt utasítás?

B. Milyen tulajdonságú pontok esetén hajtódik végre a {**}-gal jelölt utasítás?

C. Milyen tulajdonságú pontok esetén hajtódik végre a {***}-gal jelölt utasítás?

D. Lehetséges-e, hogy egy gráfban több olyan tulajdonságú pont is van, mint aminek a sorszámát a {***}-gal jelölt utasításban megadjuk? Magyarázd meg, miért!

2. feladat: Háttértár kezelés (21 pont)

Egy lemezegységen az adatok N sávban helyezkednek el, a legkülső sáv sorszáma 1. A lemezvezérlő egy adatot 1 időegység alatt képes leolvasni az aktuális sávból. Az i-edik sávból a j-edik sávba való lépéshez |i-j|+K időre van szüksége.

A fejlesztők rájöttek, hogy nem gazdaságos az olvasási kérelmeket beérkezésük sorrendjében teljesíteni. Ha például az olvasó fej az 1. sávban áll és jön egy olvasáskérés az N. sávra, akkor a vezérlő elindítja az olvasófejet befelé. Ha közben jön egy olvasáskérés az N/2-edik sávra, akkor érdemesebb lenne ott megállni, beolvasni az ott levő adatot, majd továbbmenni az N-edik sávra.

Az alábbi algoritmusok feltételezik, hogy a T tömb i-edik eleme tartalmazza, hogy pillanatnyilag hány olvasáskérés van az i-edik sávra. A T tömböt kívülről tölti fel a megfelelő időpontban egy megszakítás-kezelő eljárás.

```
Algoritmus:
  i:=1; ir:=1
  Ciklus ...
   Ha van olvasáskérés akkor
     Következő-*(i,j,ir)
     Mozgatás(i,j)
     i:=j; T(i):=0
   Elágazás vége
  Ciklus vége
Eljárás vége
```

```
Következő-A(i,j,ir):
  j1:=i; j2:=i
  Ciklus amíg T(j1)=0 és T(j2)=0
    Ha j1>1 akkor j1:=j1-1
    Ha j2<N akkor j2:=j2+1
  Ciklus vége
  Ha T(j1)>0 akkor j:=j1
          különben j:=j2
Eljárás vége.
```

```
Következő-B(i,j,ir):
  j:=i
  Ciklus amíg T(j)=0
    Ha j=1 akkor ir:=1
    Ha j=N akkor ir:=-1
    j:=j+ir
  Ciklus vége
Eljárás vége.
```

```
Következő-C(i,j,ir):
  j:=i
  Ciklus amíg T(j)=0
    Ha j=N akkor j:=1
        különben j:=j+1
  Ciklus vége
Eljárás vége.
```

A. Mi a stratégiája a három `Következő-*` eljárásnak?

B. Fogalmazd meg, milyen olvasáskéréssel kell a legtovább várni az egyes algoritmusok esetén!

C. Fogalmazd meg, milyen (nem az aktuális sávra vonatkozó) olvasáskéréssel kell a legkevesebbet várni az egyes algoritmusok esetén!

3. feladat: Szavak (26 pont)

Tekintsük az alábbi F1 és F2 függvényeket, ahol S betűket tartalmazó sorozat.

```
Függvény F1(i,j):
  Ha i>j akkor F1:=0
  egyébként ha i=j akkor F1:=1
  egyébként ha S[i]=S[j] akkor F1:=2+F1(i+1,j-1)
  egyébként F1:=Maximum(F1(i+1,j),F1(i,j-1))
Függvény vége.
```

```
Függvény F2(i,j):
  Ha i>=j akkor F2:=0
  egyébként ha S[i]=S[j] akkor F2:=F2(i+1, j-1)
  egyébként F2:=1+Minimum(F2(i+1,j),F2(i,j-1))
Függvény vége.
```

A. Mi az értéke az `F1(1,10)` függvényhívásnak, ha `S='elvermelve'`?

B. Mit számít ki az `F1(1,N)` függvényhívás, ha S pontosan N betűből áll?

C. Mi az értéke az `F2(1,10)` függvényhívásnak, ha `S='elvermelve'`?

D. Mit számít ki az `F2(1,N)` függvényhívás, ha S pontosan N betűből áll?

4. feladat: Autó rendezés (13 pont)

Egy autó parkolóban N darab autó parkol egymás mellett. Az autókat érkezési sorszámokkal azonosítjuk 1-től N-ig. Rendezni kell az autókat úgy, hogy sorszámuk szerint növekvően legyenek a parkolóban. A rendezést három dolgozó végzi a következő eljárás szerint. Egy menetben mindegyik dolgozó legfeljebb egy autót mozgathat, de csak olyan helyre, ahonnan ugyanebben a menetben egy másik autót elvisz egy dolgozó. A cél az, hogy a lehető legkevesebb menetben elvégezzék a rendezést.

Példa:

`Ha 6 autó van, az autók kezdeti sorrendje 3, 4, 1, 5, 6, 2, akkor 3 menet kell.`

A. Legkevesebb hány menet kell a rendezéshez, ha 11 autó kezdeti sorrendje:
`2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1`

B. Legkevesebb hány menet kell a rendezéshez, ha 20 autó kezdeti sorrendje:
`5, 1, 3, 7, 2, 8, 4, 10, 11, 9, 6, 20, 19, 18, 12, 15, 17, 16, 14, 13`

C. N autó esetén legrosszabb esetben hány menet kell a rendezéshez?

5. feladat: Jelek (22 pont)

Csillagászok naponta észlelnek két távoli égitestről érkező jelsorozatokat. Minden jelsorozat egy bitsorozattal azonosítható, és az alábbi szabályokkal jellemezhető.

Az A égitest esetén:

A1. Az egyetlen **0**-t tartalmazó jelsorozat észlelt.

A2. Ha **α** észlelt, akkor az **α0** és **α10** is észlelt.

A3. Minden észlelt jelsorozat megkapható az A1. és A2. szabályok véges sokszori alkalmazásával.

A B égitest esetén:

B1. Az egyetlen **0**-t és egyetlen **1**-et tartalmazó jelsorozat észlelt.

B2. Ha **β** észlelt, akkor a **0β0** és az **1β1** is észlelt.

B3. Minden észlelt jelsorozat megkapható az B1. és B2. szabályok véges sokszori alkalmazásával.

Melyek azok a jelsorozatok, amelyek mindkét égitestről származhatnak? Add meg a jelsorozatok szabályait.

2001. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Automata (25 pont)

Egy automatát a J, B, E betűk sorozatával vezérelhetünk. Az automata kezdetben az (x,y) = (0,0) koordinátájú pontban áll, és északi irányba néz. A J hatására jobbra fordul 90 fokkal, a B hatására pedig balra. Az E hatására 1 egységet előre lép az aktuális irányban. x értéke kelet felé nő, nyugat felé csökken, y értéke észak felé nő, dél felé csökken. x és y negatív számok is lehetnek.

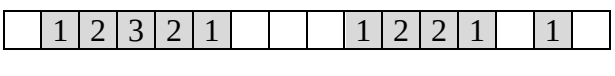
Készíts programot, amely beolvas egy legfeljebb 100 karakterből álló utasítássorozatot, majd megadja, hogy ennek hatására az automata milyen koordinátájú pontra jutott, és éppen merre néz!

Példa:

```
Bemenet: EJEEE
Kimenet: Hely: (3,1), irány: kelet
```

2. feladat: Kolbász (24 pont)

A kolbászgyárban a kolbászokat egy cső alakú tárolóban helyezik el. Az ábrán a szürke (számozott) mezők jelzik a kolbászokat, a fehérek pedig az üres helyeket. A két szélső mező biztosan üres.



A tárolót ellepték az egerek, mert különösen kedvelik a kolbászokat. Az egerek egy időegység alatt egy-egy mezőnyit tudnak lerágni minden kolbász mindkét végéről. (Az ábrába számokat írtunk, ezek azt jelzik, hogy melyik kolbászdarabot hányadik időegységben eszik meg az egerek.)

Írj programot, amely megadja, hogy mennyi idő alatt eszik meg az egerek az összes kolbászt!

A program először olvassa be a tároló hosszát (1≤N≤100) (az ábrán látható példában N=16), majd az egyes mezők leírását N darab számjeggyel! A 0 azt jelenti, hogy az i-edik mezőben nincs, az 1 pedig azt, hogy van kolbász. (A fenti ábra esetén a következő 16 számjegyet kell beírni: 0111110001111010.) Ezután ki kell írni, hogy hány időegység alatt fogy el az összes kolbász!

Példa:

```
Bemenet:  16
          0111110001111010
Kimenet:   3
```

3. feladat: Számzár (26 pont)

Pisti táskáján olyan számzár van, amely N tárcsából áll. A tárcsák különböző nagyságúak, így az egyes tárcsákon különböző számok közül lehet választani. (Pl. két tárcsa közül az egyiken 0-tól 9-ig haladnak a számok, a másikon pedig csak 0-tól 5-ig.) Sajnos, Pisti elfelejtette a kódot.

Írj programot, amely választ ad az alábbi kérdésekre!

A. A tárcsák száma (1≤N≤5) és az egyes tárcsákon előforduló számjegyek M száma (1≤M≤10) alapján legfeljebb hány kombinációt kell Pistinek kipróbálnia a táska kinyitásához?

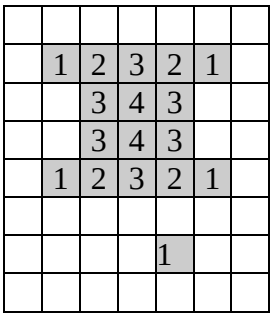
B. Legfeljebb hány próbát kell tennie Pistinek abban az esetben, ha emlékszik arra, hogy az egyik tárcsa valahogyan elromlott, azaz minden lehetséges számot elfogadott? Arra Pisti sajnos nem emlékszik, hogy melyik a hibás tárcsa.

C. Maximálisan hány kombinációt kell kipróbálnia Pistinek azon a táskán, amelyikről azt tudja, hogy a kódban van legalább két azonos számjegy?

Kilencedik-tizedik osztályosok

1. feladat: Jégtömb (20 pont)

Jégtömböt írunk le egy táblázat segítségével. A szürkével jelölt (számozott) pontok jelölik a jégtömbhöz tartozó pozíciókat. Ha a jégtömbre meleg levegőt fújunk, akkor a szélén olvadni kezd, s a keletkezett víz elfolyik.



Az olvadás szabálya: egy időegység alatt abban a mezőben levő jég olvad el, amelynek négy oldalszomszédja közül legalább kettőben levegő volt. (Ilyenek az ábrán az 1-es számmal jelölt mezők.) Ennek eredményeképpen keletkezhetnek újabb ilyen tulajdonságú mezők (az ábrán 2-vel jelöljük őket), amelyek majd a 2. időegységben olvadnak el, és így tovább.

Készíts programot, amely egy adott jégtömbre megadja, hogy hány időegység alatt olvad el teljesen, illetve időegységenként megadja, hogy a jégtömb még hány mezőből áll!

A JEGTOMB.BE állomány első sorában két egész szám van, a táblázat sorainak (1≤N≤100) és oszlopainak (1≤M≤100) a száma. A következő N sor mindegyike M számjegyet tartalmaz, közülük az i-edik sor j-edik számjegye 0, ha a táblázat i-edik sorának j-edik oszlopában levegő van, és 1, ha jég. A táblázat szélső soraiban és oszlopaiban biztosan levegő van.

A JEGTOMB.KI állomány első sorába azon időegységek T számát kell írni, amelyek elteltével a jégtömb teljesen elolvad! A következő T sorba egy-egy számot kell írni, közülük az i-edik a jéggel teli mezők száma legyen az i-edik időegység kezdetén! (A legutolsó időegység után már 0 a jéggel teli mezők száma, ezt az állományba nem szabad kiírni!)

Példa: (az ábrán látható jégtömbre)

JEGTOMB.BE	JEGTOMB.KI
8 7	4
0000000	17
0111110	12
0011100	8
0011100	2
0111110	
0000000	
0000100	
0000000	

2. feladat: Licit (20 pont)

Nevesincs város polgármestere elhatározta, hogy értékesíti a város mellett levő téglalap alakú földdarabot. A földet egyforma méretű parcellákra osztotta.

A polgármester úgy döntött, hogy a parcellákat nyilvános pályázat keretében adja el, azaz egy adott határidőig minden érdeklődő lezárt borítékban leadhatja ajánlatát. Egy pályázó csak egy ajánlatot nyújthat be, amelyben meg kell adnia, hogy melyik parcellától melyik parcelláig terjedő részt kívánja megvenni, és mennyiért.

A pályázat sikeres volt, a határidő lejártáig N pályázat érkezett. Ezek közül ki kell választani azokat az ajánlatokat, amelyek a legtöbb bevételt eredményezik, s persze úgy, hogy egyetlen parcellát sem ítélnek oda egynél több pályázónak. Egy-egy pályázó vagy az összes kért parcellát megkapja, vagy egyet sem kap meg. Előfordulhat, hogy a maximális bevétel eléréséhez nem kell eladni az összes parcellát.

Írj programot, amely megadja a választ a polgármester problémájára!

A LICIT.BE állomány első sorában a pályázatok száma ($1 \le N \le 100$) és a parcellák száma ($1 \le M \le 100$) található. A következő N sor az egyes pályázók adatait tartalmazza, a pályázókat ez a sorrend azonosítja. Mindegyik sorban 3 szám van: A B FT, ami azt jelenti, hogy a pályázó az A sorszámú parcellától ($1 \le A \le M$) a B sorszámú parcelláig ($A \le B \le M$) terjedő részért FT forintot fizetne ($1000 \le FT \le 1\,000\,000$).

A LICIT.KI állomány első sorába az elérhető legnagyobb bevételt kell írni! A második sorba a nyertes pályázók sorszámai kerüljenek egy-egy szóközzel elválasztva! Ha több megoldás is van, csak egyet kell kiírni (bármelyiket)!

Példa:

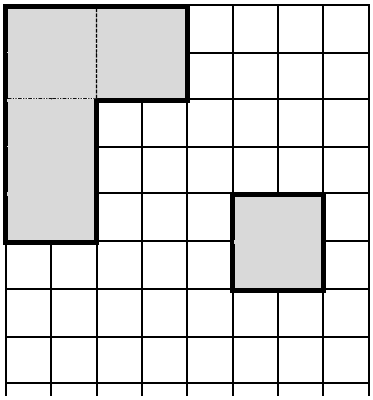
LICIT.BE	LICIT.KI
4 5	11000
1 5 10000	2 4
2 3 5000	
4 5 5000	
4 4 6000	

3. feladat: Térkép (20 pont)

Újlandia térképének (N sorból, M oszlopból álló téglalap) K db téglalap alakú részéről készítettünk nagyításokat. A téglalapok oldalai párhuzamosak a térkép oldalaival, s mindegyiket a bal felső és a jobb alsó sarkának koordinátájával adjuk meg. Az egyes téglalapok átfedhetik egymást. (Minden

koordináta 1 blokkot jelöl a térképen, azaz az (1,1) és (1,1) koordinátákkal megadott téglalap kerülete 4, területe 1 egység. A teljes térkép területe N*M, a kerülete pedig 2*N+2*M egység.)

Készíts programot a nagyított részek együttes kerületének és területének meghatározására!



A TERKEP.BE állomány első sorában N (1≤N≤160), M (1≤M≤400) és K (1≤K≤100) értéke van. A következő K sorban az egyes téglalapok leírása következik. Mindegyik sor 4 egész számot tartalmaz, a téglalapok bal felső (1≤BFY≤N, 1≤BFX≤ M) és jobb alsó (BFY≤JAY≤N, BFX≤JAX≤M) sarkának koordinátáit. A koordinátát tehát (sor, oszlop) párként adjuk meg.

A TERKEP.KI állomány egyetlen sorába a nagyított rész együttes kerületét és területét kell írni!

Példa:

```
TERKEP.BE           TERKEP.KI
10 8 3              26 18
5 6 6 7
1 1 2 4
1 1 5 2
```

4. feladat: Jelek (15 pont)

A csillagászok naponta észlelnek távoli égitestekről érkező jelsorozatokat. Minden jelsorozat egy-egy bitsorozattal azonosítható. Megfigyelték, hogy az **A** és a **B** égitestről érkező jelek nagyon szabályosak.

Az **A** égitest esetén:

A1. A **0** és a **01** két észlelt jelsorozat.

A2. Ha az **U** észlelt jelsorozat, akkor az **U0U1** és az **U1U0** is észlelt jelsorozat.

A3. Minden észlelt jelsorozat megkapható az A1. és A2. szabályok véges számú többszöri alkalmazásával.

A **B** égitest esetén:

B1. A **01** egy észlelt jelsorozat.

B2. Ha az **U** észlelt jelsorozat, akkor a **0U1** is észlelt jelsorozat.

B3. Ha az **U** és a **V** észlelt jelsorozatok, akkor az **UV** is észlelt jelsorozat.

B4. Minden észlelt jelsorozat megkapható a B1., B2. és B3. szabályok véges számú többszöri alkalmazásával.

Készíts programot, amely meghatározza, hogy a vizsgált jelsorozatok melyik égitestről érkezhettek!

A JELEK.BE állomány első sorában a vizsgálandó jelsorozatok száma (1≤N≤100) van. A következő N sor mindegyikében egy-egy jelsorozat van, amelynek hossza nem nagyobb, mint 255, és csak a 0 és az 1 karaktereket tartalmazhatja.

A JELEK.KI állományba pontosan N sort kell írni! Az i-edik sor tartalma az i-edik vizsgált jelsorozathoz tartozó válasz, amely egy vagy két karakterből álló szöveg:

- A, ha a jelsorozat csak az **A** égitestről érkezhetett,
- B, ha a jelsorozat csak a **B** égitestről érkezhetett,

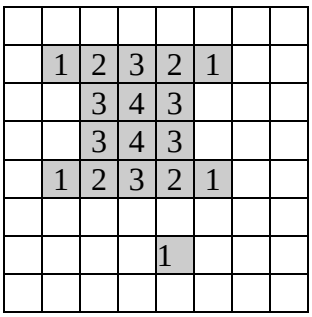
- AB, ha a jelsorozat az **A** és a B égitestről egyaránt érkezhetett,
- C, ha a jelsorozat sem az **A,** sem a **B** égitestről nem érkezhetett.

Példa:

JELEK.BE	JELEK.KI
4	A
0100	C
101011	B
000111	AB
010011	

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Jégtömb (20 pont)



Jégtömböket írunk le egy táblázat segítségével. A szürkével jelölt pontok jelölik a jégtömbbe tartozó pozíciókat. Ha a jégtömbre meleg levegőt fújunk, akkor a szélén olvadni kezd, s a keletkezett víz elfolyik.

Az olvadás szabálya: egy időegység alatt abban a mezőben levő jég olvad el (s tűnik el a táblázatból), amelynek 4 oldal-szomszédja közül legalább 2 levegő volt. (Ilyenek az ábrán 1-es számmal jelölt pontok.) Ezután keletkezhetnek újabb ilyen tulajdonságú pontok (az ábrán 2-vel jelöljük őket). amelyek a 2. időegységben olvadnak el és így tovább.

Általánosítsuk ezt a feladatot 3 dimenziósra, álljon a táblázat K darab ilyen egymás fölötti síkból (azaz a négyzetek helyett kockákkal dolgozunk)! Ekkor a mezőben levő jég akkor olvad el, ha a mező 6 lapszomszédja közül legalább 3-ban levegő van.

Készíts programot, amely egy adott 3-dimenziós jégtömbre megadja, hogy hány időegység alatt olvad el teljesen, illetve időegységenként megadja, hogy a jégtömb még hány mezőből áll!

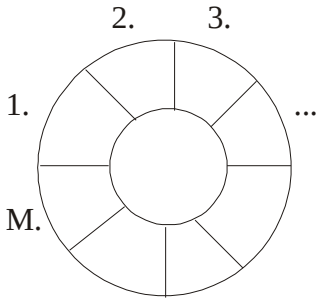
A JEGTOMB.BE állomány első sorában három egész szám van, a „táblázat" sorainak ($1 \le N \le 40$), oszlopainak ($1 \le M \le 40$) és síkjainak ($1 \le K \le 40$) a száma. Ezt követi a K darab sík leírása. Minden síkhoz N sor tartozik, amelyek mindegyike M számjegyet tartalmaz, közülük az i-edik sor j-edik számjegye 0, ha az adott sík i-edik sorának j-edik oszlopában levegő van, és 1, ha jég. A "táblázat" szélső mezőiben biztosan levegő van.

A JEGTOMB.KI állomány első sorába azon időegységek T számát kell írni, ami alatt a jég teljesen elolvad! A következő T sorba egy-egy számot kell írni, közülük az i-edik a jéggel teli mezők száma legyen az i-edik időegység kezdetén! (A legutolsó időegység után már 0 a jéggel teli mezők száma, ezt az állományba nem szabad kiírni!)

Példa:

JEGTOMB.BE		JEGTOMB.KI
4 5 4		2
00000		12
00000		4
00000		
00000	itt az 1. lap vége	
00000		
01110		
01110		
00000	itt a 2. lap vége	
00000		
01110		
01110		
00000	itt a 3. lap vége	
00000		
00000		
00000		
00000	itt a 4. lap vége	

2. feladat: Licit (20 pont)



Nevesincs sziget polgármestere elhatározta, hogy értékesíti a sziget tengerpartját. A partot egyforma méretű parcellákra osztotta.

A polgármester úgy döntött, hogy a parcellákat nyilvános pályázat keretében adja el, azaz egy adott határidőig minden érdeklődő lezárt borítékban leadhatja ajánlatát. Egy pályázó csak egy ajánlatot nyújthat be, amelyben meg kell adnia, hogy melyik parcellától melyik parcelláig terjedő részt kívánja megvenni, és mennyiért.

A pályázat sikeres volt, a határidő lejártáig N pályázat érkezett. Ezek közül ki kell választani azokat az ajánlatokat, amelyek a legtöbb bevételt eredményezik, s persze úgy, hogy egyetlen parcellát sem ítélünk oda egynél több pályázónak. Egy-egy pályázó vagy az összes kért parcellát megkapja, vagy egyet sem kap meg. Előfordulhat, hogy a maximális bevétel eléréséhez nem kell eladni az összes parcellát.

Írj programot, amely megadja a választ a polgármester problémájára!

A LICIT.BE állomány első sorában a pályázatok száma ($1 \le N \le 100$) és a parcellák száma ($1 \le M \le 100$) található. A következő N sor az egyes pályázók adatait tartalmazza, a pályázókat ez a sorrend azonosítja. Mindegyik 3 számot tartalmaz: A B FT, ami azt jelenti, hogy a pályázó az A sorszámú parcellától ($1 \le A \le M$) a B sorszámú parcelláig ($1 \le B \le M$) terjedő részért FT forintot fizetne ($1000 \le FT \le 1\ 000\ 000$). Ha $B < A$, akkor a pályázó B-től M-ig és 1-től A-ig szeretné megvásárolni a parcellákat.

A LICIT.KI állomány első sorába az elérhető legnagyobb bevételt kell írni! A második sorba azon pályázók sorszámai kerüljenek egy-egy szóközzel elválasztva, akik nyerése esetén érhető el a legnagyobb bevétel! Ha több megoldás is lenne, akkor közülük csak egyet kell kiírni (bármelyiket)!

Példa:

```
LICIT.BE            LICIT.KI
5 6                 14000
1 5 10000           2 4 5
2 3 5000
4 5 5000
4 4 6000
6 1 3000
```

3. feladat: Hálózat (15 pont)

Egy hírközlési hálózat csomópontokból és csomópont-párokat összekötő vezetékekből áll. Egy csomópont-pár tagjait közvetlenül összekötő vezeték kétirányú kapcsolatot tesz lehetővé a két pont között. E két csomópont közötti adatátvitel sebességét a vezeték sávszélességének nevezzük. Két adott pont között az adatátvitelt közvetlennek nevezzük, ha a két pont össze van kötve vezetékkel, és közvetettnek, ha az adatok közbeiktatott csomópontokon is áthaladnak. A két tetszőleges pont közötti útvonal átviteli sebessége a közbeiktatott vezetékek sávszélességének minimuma. Bármely két csomópont között legfeljebb egy vezeték van, azonban több közvetett összeköttetés is létezhet közöttük.

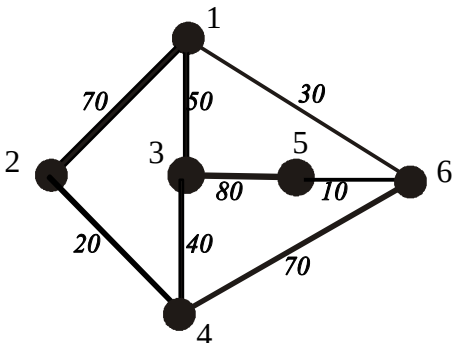
Készíts olyan programot, amely kiszámítja a hálózat két adott csomópontja között a legnagyobb adatátviteli sebességet nyújtó útvonal sávszélességét!

A HALOZAT.BE állomány első sora a csomópontok számát ($2 \le N \le 100$), és a két kijelölt csomópont sorszámát ($1 \le A \ne B \le N$) tartalmazza. A második sorban a közvetlenül összekötött csomópont-párok száma ($1 \le M \le 10\,000$) van. A következő M sor mindegyikében három egész szám van a két közvetlenül összekötött csomópont azonosítója ($1 \le X \ne Y \le N$), valamint az X-et az Y-nal összekötő vezeték sávszélessége ($1 \le S \le 1000$).

A HALOZAT.KI állomány egyetlen sorába egy egész számot kell írni: a kijelölt A és B csomópontok közötti lehető legnagyobb sávszélességet! Ha a két csomópont között nincs út, akkor 0-t kell kiírni!

Példa:

```
HALOZAT.BE          HALOZAT.KI
6 1 6               40
8
1 2 70
1 3 50
1 6 30
2 4 20
3 4 40
3 5 80
4 6 70
5 6 10
```



4. feladat: Gén (20 pont)

A genetikai kód egyértelműen leírható a négyféle bázis (adenin, citozin, guanin, timin) sorrendjével. A bázisokat a kezdőbetűjükkel (A,C,G,T) jelöljük. Óriásmolekulák bázisszekvenciáját nehéz meghatározni, ezért a vizsgálat előtt az óriásmolekulát megfelelő enzimekkel feldarabolják kisebb (rövidebb) molekuladarabokra. Ugyanazt az óriásmolekulát többféleképpen is feldarabolják, azt azonban nem lehet tudni, hogy a keletkező kisebb molekulák melyik darabolásból származnak. Tudjuk viszont, hogy (1) a molekuladarabok különböznek egymástól, (2) minden molekuladarab csak egy

helyre illeszthető be az óriásmolekulába, (3) ugyanazon a helyen legfeljebb egyszer vágjuk el az óriásmolekulát.

A feldarabolás után kémiai módszerekkel meghatározzák a keletkezett molekuladarabok bázissorrendjét.

Az egyes darabok csak a leírt sorrendben használhatók fel, megfordítani nem szabad őket!

Készíts programot, amely a molekuladarabok bázissorrendjéből előállítja az eredeti óriásmolekula bázissorrendjét!

A GEN.BE állomány első sorában a feldarabolások száma (2≤K≤10) és egy szóközzel elválasztva a keletkezett molekuladarabok száma (1≤N≤100) található. A következő N sor mindegyike egy 10-25 karakterből álló betűsorozatot tartalmaz, az egyes molekuladarabok bázissorrendjét.

A GEN.KI egyetlen sorába egy legfeljebb 250 karakterből álló szöveget kell írni, az eredeti óriásmolekula bázissorrendjét! Ha több megoldás is lenne, akkor közülük csak egyet kell kiírni (bármelyiket)!

Példa: (a karaktersorozatok hossza itt az érthetőség miatt kisebb, mint a feladatban megadott minimális határ)

```
GEN.BE                  GEN.KI
2 5                     AACGAGT
AACG
AAC
AGT
GA
GT
```

2001. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Péntek (21 pont)

Egyesek nagyon szerencsétlen napnak tartják, ha a pénteki nap valamelyik hónap 13. napjára esik.

Készíts programot, amely egy adott évben megmondja, hogy mely hónapok 13. napja esik péntekre!

A program első lépésként olvassa be, hogy melyik évről van szó, illetve, hogy az év első napja milyen napra esik (HÉTFŐ vagy KEDD vagy … vagy VASÁRNAP). Ezután írja ki azon hónapok nevét, amelyek 13. napja pénteken van! Figyelj a szökőévekre!

Példa: `Bemenet: 2001 HÉTFŐ`

`Kimenet: ÁPRILIS JÚLIUS`

2. feladat: Hegy (27 pont)

Egy hegymászó a tervezett útvonala mentén méterenként megmérte a felszín tengerszint feletti magasságát. N helyen kapott mérési adatokat. Emelkedőnek nevezzük azt a számsorozatot, amelynek minden eleme nagyobb, mint az előtte levő. Az emelkedő helye az ilyen számsorozat első és utolsó tagjának sorszáma, a hossza pedig a számsorozatban levő számok darabszáma. (Emelkedő lehet balról jobbra, illetve jobbról balra haladva is!)

Készíts programot, amely megadja, hogy az út során hol volt a leghosszabb emelkedő! (Ha több egyforma van, közülük egyet kell megadni.) Ha nincs emelkedő az út során, akkor írja ki, hogy NINCS!

A feladat megoldásához először be kell olvasni a mérések számát (1≤N≤100), majd pedig az N darab mérést; majd ezután ki kell írni az eredményt.

Példa:

```
Bemenet:N=10
        Mérések: 100,110,115,110,105,115,125,130,125,125
Kimenet: 5,8
```

3. feladat: Bob (27 pont)

A négyes bobban minden sportolónak más a feladata: van *kormányos*, *második*, *harmadik*, *indító*. A csapathoz tartozik még egy *tartalék* is. Egy igazságos edző úgy osztja be a csapatot, hogy mindenki nagyjából ugyanannyiszor szerepeljen minden poszton, azaz a versenyzők azonos poszton való indulásainak száma legfeljebb eggyel térjen el.

Írj programot, ami a versenyzők szezonbeli beosztása alapján eldönti, hogy igazságos volt-e eddig az edző, majd megadja posztonként azokat, akik a következő versenyen azt adott poszton indíthatók, s emiatt az igazságosság elve nem sérül!

A program először olvassa be a fordulók F számát (1≤F≤10), majd az egyes fordulók beosztását: *kormányos*, *második, harmadik*, *indító*, *tartalék* sorrendben (ez F*5 név).

A beolvasás után a program ellenőrizze, hogy eddig igazságos volt-e az edző! Ha az edző igazságos volt, akkor ezután írja ki (tetszőleges sorrendben), hogy az igazságosság miatt ki lehet a következő versenyen *kormányos, második, harmadik*, illetve *indító*.

Példa:

```
Fordulók száma: 7
1. forduló: Ali     Béla    Csaba   Dani    Elek
2. forduló: Béla    Csaba   Dani    Elek    Ali
3. forduló: Csaba   Dani    Elek    Ali     Béla
4. forduló: Dani    Elek    Ali     Béla    Csaba
5. forduló: Elek    Ali     Béla    Csaba   Dani
6. forduló: Ali     Elek    Béla    Csaba   Dani
7. forduló: Elek    Ali     Dani    Béla    Csaba
IGAZSÁGOS
Kormányos lehet:    Béla Csaba Dani
Második lehet:      Béla Csaba Dani
Harmadik lehet:     Ali Csaba Elek
Indító lehet:       Ali Dani Elek
```

Kilencedik-tizedik osztályosok

1. feladat: Terem (16 pont)

Iskolád alapításának évfordulóján nagyszabású ünnepséget szervez. Egy napra sok eseményt tervez. Kiderült, hogy lesznek események, amelyek részben egy időben zajlanak. Ezért meg kell határozni, hogy legkevesebb hány termet kell előkészíteni ahhoz, hogy minden esemény számára legyen terem foglalva, és természetesen az események ne ütközzenek. Minden betervezett eseménynek ismerjük a kezdési és befejezési időpontját, amit percben adtak meg. Ha egy esemény az A perctől a B percig tart, akkor ugyanabba a terembe beosztott bármely másik esemény vagy A-nál korábban véget ér, vagy B-nél később kezdődhet.

Készíts programot, amely kiszámítja, hogy legkevesebb hány terem kell ahhoz, hogy minden betervezett eseményt meg lehessen tartani, továbbá megad egy lehetséges terembeosztást!

A TEREM.BE állomány első sorában az események száma van (1≤N≤1000). A következő N sor mindegyikében két egész szám, A és B van. Az A szám egy esemény kezdő, a B pedig ugyanezen esemény befejező időpontja (1≤A≤B≤1440).

A TEREM.KI állomány első sorába az összes esemény beosztásához szükséges legkevesebb terem T számát kell írni! A következő T sorban kell megadni a termek beosztását! Egy sorba azon események sorszámait kell írni egy-egy szóközzel elválasztva, amelyek ugyanazon teremben lesznek megtartva (a különböző sorokban lévők pedig mind külön teremben)!

Példa:

TEREM.BE	TEREM.KI
8	4
1100 1200	1 2 4 8
500 520	3 5
510 570	6
600 630	7
630 700	
700 800	
600 800	
650 700	

2. feladat: Túra (18 pont)

Magyarország folyóiról feljegyeztük, hogy milyen másik folyóba folynak bele (pl. a Rába a Dunába, a Sajó a Tiszába, a Hernád a Sajóba, …). Minden folyó legfeljebb 1 másikba folyhat bele, de lehet hogy egybe sem (pl. Duna, de a Zala sem folyóba folyik bele).

Csónaktúrákat szeretnénk szervezni, de a könnyebbség kedvéért csak úgy, hogy minden folyón a folyás irányában haladjunk.

Készíts programot, amely megadja, hogy

A. két különböző folyón indult túra hol találkozhat;

B. az első túrát bevárhat-e egy második úgy, hogy nem indul el addig, amíg az első oda nem ér, és ha igen, akkor az elsőnek hány folyón kell addig haladnia (ha ugyanazon a folyón indulnak, akkor 1, ha az egyik folyó éppen belefolyik a másikba, akkor 2, …).

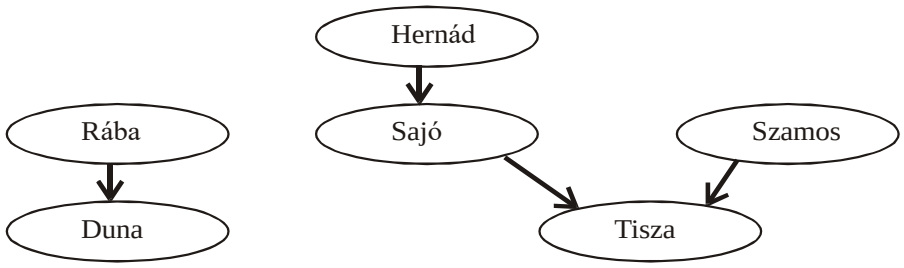
A TURA.BE állomány első sorában az az N egész szám van, ahány folyóról tudjuk, hogy melyikbe folyik bele (1≤N≤1000). A következő 2*N sor mindegyike egy-egy folyó nevét tartalmazza, a második, negyedik, ... sor azt hogy melyik folyó, a harmadik, ötödik, … pedig azt, hogy melyikbe folyik bele. Az utolsó két sorban egy-egy folyó neve van, az első és a második túra kezdete. (A folyónevek legfeljebb 20 betűsek.)

A TURA.KI állomány első sorába annak a folyónak a nevét kell írni, ahol a két túra először találkozhat, a sor legyen üres, ha a két túra Magyarországon nem találkozhat (pl. ha az első a Dunán, a második pedig a Tiszán indult)! A második sorba azt az egész számot kell írni, ahány folyón az első túrának át kell haladnia, hogy a második túra kezdetéhez érjen! Ha a második nem tudja bevárni az elsőt, akkor ez a szám 0 legyen!

Példa:

```
TURA.BE
4
Rába
Duna
Hernád
Sajó
Szamos
Tisza
Sajó
Tisza
Hernád
Szamos
```

```
TURA.KI
Tisza
0
```



3. feladat: Piramis (20 pont)

A piramisépítők egy négyzet alapú területre építik a piramist. A terület minden egységnyi négyzetére adott darabszámú kőkockát helyeznek. Amikor egy újabb követ kell elhelyezni, akkor valahonnan a piramis széléről indulnak, és úgy haladnak, hogy minden lépésben pontosan eggyel magasabb szomszéd helyre lépnek. (A szomszédos hely átlósan nem lehet!) A követ mindig oda teszik, ahonnan nem tudnak szomszédos helyre továbblépni.

Készíts programot, amely megadja a leghosszabb utat, amelyen a piramisépítők egy követ elvihetnek a helyére!

A PIRAMIS.BE állomány első sorában a piramist tartalmazó négyzet oldalhossza van ($1 \le N \le 100$). A következő N sor mindegyike N egész számot tartalmaz, az egyes pozíciókban levő kőkockák számát ($0 \le$ darabszám $\le 10\,000$).

A. PIRAMIS KI állomány első sorába a leghosszabb út hosszát kell írni (azon lépések számát, ahány lépés alatt egy tetszőleges kezdőpozícióból szomszéd helyeken át egyesével lehet felfelé lépkedni), a második sorba pedig az ehhez az úthoz tartozó kezdő pozíció sor- és oszlopindexét! Ha sehonnan sem lehet lépni, akkor az első sorba 0, a második sorba tetszőleges pozíció írandó!

Példa:

```
PIRAMIS.BE
6
1 2 2 2 2 2
4 3 4 2 2 1
1 1 5 6 1 8
1 1 1 9 6 7
1 3 4 4 5 1
1 2 1 1 1 1
```

```
PIRAMIS.KI
5
1 1
```

Itt a bal felső sarokból indulva pontosan 5 lépést lehet megtenni. Más helyről indulva ennél csak kevesebbet lehet.

4. feladat: Fazekas (21 pont)

Egy fazekas műhelyében sorban várakoznak az kiégetésre váró tárgyak. Minden tárgyról tudjuk, hogy mennyi az a legkevesebb idő, ami a kiégetéséhez kell. Az égetésre váró tárgyakat az érkezésük sorrendjében kell kiégetni. Egyszerre több tárgyat is rakhatunk a kemencébe, azonban legfeljebb annyit, amennyi a kemence adott kapacitása. Az égetési idő egy menetben mindig a kemencébe rakott tárgyak minimális égetési idejének a maximuma kell legyen.

Készíts programot, amely kiszámítja, hogy legkevesebb mennyi idő kell az összes tárgy kiégetéséhez, továbbá megadja azt is, hogy ezen idő eléréséhez mely tárgyakat kell egy-egy menetben a kemencében együtt égetni!

A FAZEKAS.BE állomány első sora két egész számot tartalmaz; a tárgyak számát (1≤N≤10 000) és a kemence kapacitását (1≤K≤100). A következő N sor mindegyike egy pozitív egész számot tartalmaz; a tárgy minimális égetési idejét, ami nem nagyobb, mint 200.

A FAZEKAS.KI állomány első sorába az összes tárgy kiégetéséhez minimálisan szükséges időt kell írni! A következő sorok mindegyikébe két egész számot, I-t és J-t kell írni egy szóközzel elválasztva, I az első, J pedig az utolsó tárgy sorszáma, amelyek egyszerre kerülnek a kemencébe!

Példa:

FAZEKAS.BE	FAZEKAS.KI
7 3	75
10	1 2
8	3 4
20	5 7
25	
30	
12	
40	

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Folyók (10 pont)

Magyarország folyóiról feljegyeztük, hogy milyen másik folyóba folynak bele (pl. a Rába a Dunába, a Sajó a Tiszába, a Hernád a Sajóba, …). Minden folyó legfeljebb 1 másikba folyhat bele, de lehet hogy egybe sem (pl. Duna, de a Zala sem folyóba folyik bele).

A folyók szennyezettségének megállapításához szükség van arra, hogy megállapítsuk, melyik folyóba honnan érkezhet *szennyezett* víz.

Készíts programot, amely megadja, hogy

A. egy folyóba mely folyókból érkezhet víz (akár másik folyón keresztül is);

B. egy folyóba került szennyezés mely más folyókat szennyezhet meg!

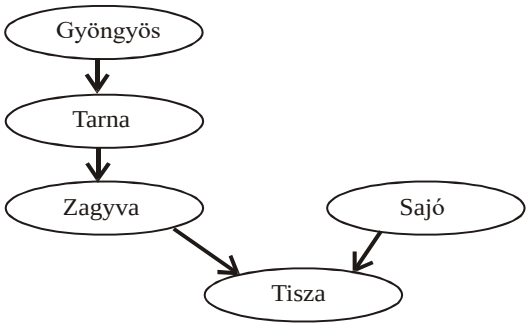
A FOLYO.BE állomány első sorában az az N egész szám van, ahány folyóról tudjuk, hogy melyikbe folyik bele (1≤N≤1000). A következő 2*N sor mindegyike egy-egy folyó nevét tartalmazza, a második, negyedik, ... sor azt hogy melyik folyó, a harmadik, ötödik, … pedig azt, hogy melyikbe folyik bele. Az utolsó sorban egy folyó neve van, az, amelyről az A és a B kérdés szól. (A folyónevek legfeljebb 20 betűsek.)

A FOLYO.KI állomány első sorába azoknak a folyóknak a K számát kell írni, amelyekből érkezhet víz a FOLYO.BE állomány utolsó sorában levő folyóba! A következő K sorba kell kiírni soronként a folyók nevét! A K+1-edik azoknak a folyóknak az L számát kell írni, amelyekbe eljuthat a szennyeződés a megadott folyóból! Az ezt követő L sorba kell kiírni soronként a folyók nevét!

Példa:

FOLYO.BE
```
4
Zagyva
Tisza
Sajó
Tisza
Tarna
Zagyva
Gyöngyös
Tarna
Zagyva
```

FOLYO.KI
```
2
Tarna
Gyöngyös
1
Tisza
```



2. feladat: Kemence (15 pont)

A porcelángyár égetőkemencéjéhez futószalagon érkeznek az égetésre váró tárgyak. Minden tárgynak ismert a súlya és a kiégetéséhez minimálisan szükséges idő. Az égetésre váró tárgyakat az érkezésük sorrendjében kell kiégetni. Egyszerre több tárgyat is rakhatunk a kemencébe, az összsúlyuk azonban nem haladhatja meg a kemence kapacitását. Az égetési idő egy menetben mindig a kemencébe rakott tárgyak minimális égetési idejének a maximuma kell legyen.

Készíts olyan programot, amely kiszámítja, hogy legkevesebb mennyi idő kell az összes tárgy kiégetéséhez, továbbá megadja azt is, hogy ezen idő eléréséhez mely tárgyakat kell egy-egy menetben a kemencében együtt égetni!

A KEMENCE.BE állomány első sorában a tárgyak száma ($1 \le N \le 10\,000$) és a kemence kapacitása ($1 \le K \le 1000$) van. A következő N sor mindegyike két egész számot tartalmaz; egy tárgy súlyát ($1 \le S \le 1000$) és minimális égetési idejét ($1 \le E \le 100$).

A KEMENCE.KI állomány első sorába az összes tárgy kiégetéséhez minimálisan szükséges időt kell írni! A következő sorok mindegyikébe két egész számot, I-t és J-t kell írni, I az első, J pedig az utolsó tárgy sorszáma, amelyek egyszerre kerülnek a kemencébe!

Példa:

KEMENCE.BE
```
6 10
3 10
2 12
7 20
5 15
3 11
2 16
```

KEMENCE.KI
```
46
1 1
2 3
4 6
```

3. feladat: Hegység (20 pont)

Egy hegymászó megkapta egy hegység domborzati térképét, amely egy négyzetrács-háló egyes pontjaiban tartalmazza a felszín tengerszint feletti magasságát. A hegymászás során a hegység tetszőleges pontjáról indulhat, s minden lépésben a 4 szomszédos hely valamelyikére léphet (tehát átlósan nem). Egy emelkedő út olyan lépéssorozat, amikor minden egyes érintett hely magasabb az előzőnél, az út hossza pedig a megtett lépések száma.

Készíts programot, amely megadja a leghosszabb utat, amelyen egy hegymászó folyamatosan felfelé haladhat! Ha több megoldás is van, elég csak egyet megadni!

A HEGYSEG.BE állomány első sorában a hegység domborzati térképét tartalmazó téglalap sorai és oszlopai száma van (1≤N,M≤100). A következő N sor mindegyike M egész számot tartalmaz, az egyes pozíciók tengerszint feletti magasságát (0≤magasság≤10 000).

A HEGYSEG.KI állomány első sorába a leghosszabb út hosszát kell írni (azon lépések számát, ahány lépés alatt egy tetszőleges kezdőpozícióból szomszéd helyeken át folyamatosan lehet felfelé lépkedni), a második sorba pedig az ehhez az úthoz tartozó kezdő pozíció sor- és oszlopindexét! Ha sehonnan sem lehet lépni, akkor az első sorba 0, a második sorba tetszőleges pozíció írandó!

Példa:

HEGYSEG.BE

```
6 8
2 2 1 2 2 2 1 1
4 3 6 9 2 1 1 1
5 1 7 8 1 8 1 1
1 1 1 1 6 7 1 1
1 3 4 4 5 1 1 1
1 2 1 1 1 1 1 1
```

HEGYSEG.KI

```
6
1 3
```

Itt az 1. sor 3. helyéről indulva pontosan 6 lépést lehet megtenni. Más helyről indulva ennél csak kevesebbet lehet. A táblázatban felfedezhető 1 4 5 6 7 8 magasságú pontokon átvezető út eggyel rövidebb.

4. feladat: Vírus (30 pont)

Kutatók megfigyelték, hogy egyes vírustörzsek nagyon sok, akár több milliárd variánst is tartalmazhatnak. Az egy törzshöz tartozó vírusok felsorolása nagyon költséges lenne, azonban azt is észrevették, hogy a variánsok nagyon hasonlítanak egymásra. Ez lehetővé teszi, hogy az informatikában jól ismert módszer alkalmazásával, úgynevezett víruskifejezéssel adjuk meg az egy törzshöz tartozó vírusokat.

Minden víruskifejezés olyan jelsorozat, amely az (angol) ábécé 'A' .. 'Z' nagybetűi, a '[', ']' kezdő- és végzárójel, a '-' mínusz jel és a '¦' függőleges vonal jelekből a következő 3 szabály véges sokszori alkalmazásával kapható:

1. Minden betű ('A'-tól 'Z'-ig) víruskifejezés, és a betűvel azonosított vírust jelöli. A '-' jel is víruskifejezés, amely az üres (egyetlen betűt sem tartalmazó) jelsorozatot jelöli.
2. Ha α és β víruskifejezés, akkor $\alpha\beta$ is víruskifejezés, és pontosan azokat a vírusokat jelöli, amelyek XY alakúak, ahol X egy α által, Y pedig egy β által jelölt vírus.
3. Ha α_1 , α_2 ,…, α_k víruskifejezések (k≥2), akkor $[\alpha_1¦\alpha_2¦\ldots¦\alpha_k]$ is víruskifejezés, és pontosan azokat a vírusokat jelöli, amelyeket valamelyik α_i (i=1,…, k) alternatíva jelöl.

Például, az [A¦B¦-]D[BB¦A[B¦C]][X¦-] víruskifejezés a következő 12 vírust jelöli: ADBBX, ADBB, ADABX, ADAB, ADACX, ADAC, BDBBX, BDBB, BDABX, BDAB, BDACX, BDAC. A [A¦B][A¦B¦-]- 31 szer, víruskifejezés az összes olyan vírust jelöli, amely legfeljebb 32 betűből áll, és minden betű vagy A vagy B.

Készíts programot, amely a bemenetben adott víruskifejezés és karaktersorozatokra eldönti, hogy azok a víruskifejezés által jelölt vírustörzshöz tartoznak-e!

Az VIRUS.BE állomány első sorában egy szabályos víruskifejezés van, amelyben a betű, és - karakterek száma legfeljebb 100. A víruskifejezés után egy pont (.) karakter van. A második sorban egy egész szám, a vizsgálandó elemek M száma van (1≤M≤10). A következő M sor mindegyikében egy legfeljebb 100 karakterből álló karaktersorozat áll, amelyekről el kell dönteni, hogy a víruskifejezés által jelölt vírustőrzsbe tartoznak-e.

Az VIRUS.KI állomány pontosan M sort tartalmazzon! A megfelelő sorban a VIRUS szó legyen, ha a vizsgált karaktersorozat a vírustörzsbe tartozik, egyébként pedig az a legnagyobb K szám,

amelyre igaz, hogy van olyan vírus, amelynek első K betűje megegyezik a vizsgált jelsorozat első K betűjével!

Példa:

```
VIRUS.BE:                    VIRUS.KI:
[A¦B¦-]D[BB¦A[B¦C]][X¦-].    VIRUS
3                            1
BDAX                         4
AAAAA
BDBBY
```

A verseny végeredménye:

I. korcsoport

1. Acsai Péter	Petőfi Sándor Általános Iskola, Nagykőrös
2. Kőszegi Judit	Árpád Vezér Gimnázium, Sárospatak
Isza Péter	Dózsa György Általános Iskola, Debrecen
4. Nikházy László	Kazinczy Ferenc Gimnázium, Győr
5. Ráksi Ádám	Arany János Általános Iskola, Százhalombatta
Vincze János	DE Arany János Gyakorló Általános Iskola, Debrecen
7. Tassy Gergely	Veres Péter Gimnázium, Budapest
Láda Ákos	Radnóti Miklós Gimnázium, Dunakeszi
Margetán Zsombor	Bencés Gimnázium, Pannonhalma
10. Paulin Roland	Fazekas Mihály Gimnázium, Budapest
Soltész Zoltán	Várkonyi István Általános Iskola, Cegléd

II. korcsoport

1. Csóka Endre	Fazekas Mihály Gimnázium, Debrecen
Pelládi Gábor	Földes Ferenc Gimnázium, Miskolc
3. Gruber László	Mechwart András Gépipari és Informatikai SzKI, Debrecen
4. Simon Balázs	Révai Miklós Gimnázium, Győr
5. Németh Zsolt	Széchenyi István Közgazdasági Szakközépiskola, Százhalombatta
6. Bárkai János	Berzsenyi Dániel Gimnázium, Budapest
7. Ruppert László	Janus Pannonius Gimnázium, Pécs
8. Szabó Dávid	Kölcsey Ferenc Gimnázium, Zalaegerszeg
9. Rucz Péter	Bencés Gimnázium, Pannonhalma
10. Bergmann Gábor	Berzsenyi Dániel Gimnázium, Budapest

III. korcsoport

1. Novák Ádám	Neumann János Közgazdasági SzKI és Gimnázium, Eger
2. Pallos Péter	Fazekas Mihály Gimnázium, Budapest
3. Ekler Márton	Zrínyi Miklós Gimnázium, Zalaegerszeg
Károly Dávid	Alternatív Közgazdasági Gimnázium, Budapest
5. Micskó Viktor	Lengyel József Gimnázium, Oroszlány
6. Siroki Péter	Mechwart András Gépipari és Informatikai SzKI, Debrecen
Hargitai Gábor	Bólyai János Gimnázium, Ócsa
Ritter Ádám	Fazekas Mihály Gimnázium, Budapest
9. Gazda Tamás	Tóth Árpád Gimnázium, Debrecen
10. Szoboszlay Dániel	Radnóti Miklós Gimnázium, Budapest

2002. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Válogatós (26 pont)

Lányok (N) és fiúk (M) ülnek egy sorban, vegyesen (N+M>0). Szeretnénk úgy ültetni őket, hogy az első N széken üljenek a lányok, a következő M széken pedig a fiúk.

A feladat megoldására a következő algoritmust találtuk ki:

1. Az első széken levőt felállítjuk (kezdetben ez lesz a szabad szék).

2. Az utolsó széktől visszafelé, de legfeljebb a szabad székig haladva keresünk egy lányt.

3. Ha találunk, akkor megjegyezzük, hogy legközelebb már csak innentől kezdve kell lányt keresni, és a lányt átültetjük a szabad székre.

4. Ezután az első széktől, de legfeljebb a szabad székig haladva keresünk egy fiút.

5. Ha találunk, akkor megjegyezzük, hogy legközelebb már csak innentől kezdve kell fiút keresni, és a fiút átültetjük a szabad székre.

6. Ha egyik keresés során sem értük el a szabad széket, akkor a 2. ponttól kezdve folytatjuk a keresést.

7. A szabadon maradt székre ültetjük azt, akit az első lépésben felállítottunk.

Példa:

`(Dóra,Endre,Ágnes,István)` ⇒ `(-,Endre,Ágnes,István)` ⇒ `(Ágnes,Endre,-,István)` ⇒ `(Ágnes,-,Endre,István)` ⇒ `(Ágnes,Dóra,Endre,István)`

A. Mi lesz az eredmény, ha kezdetben a székeken ülők sorrendje (azaz a kezdő ülésrend): `(Erika,János,Nóra)`? A fenti példához hasonlóan lépésenként írd le!

B. Mi lesz az eredmény, ha a kezdő ülésrend: `(Anna,Emese,Béla,Enikő,András,Jakab,Eszter,Móni,László,Piri,Péter)`? A fenti példához hasonlóan lépésenként írd le!

C. Van olyan kezdő ülésrend, hogy csak egyetlen gyereknek kell mozognia. Melyik gyereknek, milyen kezdő ülésrend esetén?

D. Van olyan kezdő ülésrend, hogy minden gyereknek mozognia kell. Melyik ez a kezdő ülésrend?

2. feladat: Kockapóker (30 pont)

Egy kockajátékot négy dobókockával játszanak. A dobás után összeszámolják, hogy a négy kocka közül hány mutat egyforma számot.

A lehetőségek:

Egy pár: két szám egyforma, a másik kettő tőlük és egymástól is különböző (pl. 3,1,3,5).

Két pár: két-két szám egyforma, de a párok egymástól különbözők (pl. 2,4, 4,2).

Terc: három szám egyforma, a negyedik tőlük különböző (pl. 6,6,1,6).

Póker: mind a négy szám egyforma (pl. 5,5,5,5).

Semmi: mind a négy szám különböző (pl. 1,6,2,5).

Az alábbi algoritmus az A,B,C,D változók értéke alapján kiírja, hogy milyen dobásunk volt (segítség: az algoritmusban minden elágazásnak van "akkor" és van "különben" ága is). Mit kell írni a ?1 … ?15 helyébe, hogy a helyes eredményeket kapjuk meg?

```
Póker(A,B,C,D):
  Ha A=B akkor
     Ha C=D akkor  ha A=D akkor  Ki: ?1
                           különben Ki: ?2
           különben ha A=C akkor  Ki: ?3
                    különben ha B=D akkor  Ki: ?4
                                 különben Ki: ?5
  különben ha C=D akkor ha A=C akkor Ki: ?6
                           különben ha B=D akkor Ki: ?7
                           különben Ki: ?8
                  különben ha A=C akkor  ha B=D akkor  Ki: ?9
                                                 különben Ki:?10
                  különben ha A=D akkor  ha B=C akkor  Ki: ?11
                                                 különben Ki: ?12
                  különben ha B=D akkor  Ki: ?13
                                 különben ha B=C akkor  Ki: ?14
                                                 különben Ki: ?15
Eljárás vége.
```

3. feladat: Címletezés (29 pont)

Egy pénztárban hatféle bankjegyet (címletet) kezelnek. Az alábbi címletező algoritmus, szándékunk szerint, meghatározza, hogy a P összeg hogyan fizethető ki a lehető legkevesebb bankjeggyel. Az eredményt a DB vektor tartalmazza. Minden címletből korlátlan mennyiség használható fel.

```
Címletezés(P,C,DB):
  K:=6; DB:=(0,0,0,0,0,0)
  Ciklus amíg P>0
    Ciklus amíg P≥C(K)
      DB(K):=DB(K)+1; P:=P-C(K)
    Ciklus vége
    K:=K-1
  Ciklus vége
Eljárás vége.
```

Ez az algoritmus például a C=(1,2,5,10,20,50) címletkészlet és 96 forint kifizetése esetén a DB=(1,0,1,0,2,1) vektort állítja elő.

Legyenek a címletkészletek a következők:

i.	C=(1,3,6,10,60,100)	i.	C=(1,2,5,12,60,120)
ii.	C=(1,4,6,10,40,60)	ii.	C=(1,5,7,21,35,49)

A. Mi lesz a DB vektor értéke 73 forint kifizetésekor a négy címletkészlet esetén?

B. A négy címletkészlet közül melyeknél címletez hibásan az algoritmus (azaz nem a lehető legkevesebb bankjegyet használja fel)?

C. Minden hibás eredményhez vezető címletkészletre add meg azt a legkisebb P összeget, amelynek a kifizetésére az algoritmus nem a lehető legkevesebb bankjegyet használja fel! Add meg az algoritmus által előállított hibás, valamint a helyes DB vektort is!

4. feladat: Piramis (15 pont)

Dobozokból piramist építünk. A piramis minden szintjén (az alsót kivéve) pontosan kétszer annyi doboz van, mint a felette levő szinten. A dobozokat az ábra szerint sorszámozzuk.

Definíció: Az I sorszámú **doboz alatti** doboznak a 2*I és a 2*I+1 sorszámút nevezzük, az I sorszámú **doboz feletti** doboznak pedig az I div 2 sorszámút.

A következő algoritmust hajtjuk végre N darab egész számmal. Kezdetben a dobozok üresek. Az első számot az 1 sorszámú dobozba tesszük. Ha már I-1 darab számot felhasználtunk, akkor az I-edik számot az I sorszámú dobozba tesszük, majd összehasonlítjuk a felette levővel; ha nagyobb nála, akkor felcseréljük a dobozok tartalmát. A csere után a számot ismét összehasonlítjuk a felette levővel, és ha nagyobb nála, ismét cserélünk és így tovább.

A. Add meg a piramis állapotát (azaz a dobozok tartalmát) minden lépés után, ha a bemenő számsorozat: (7,2,9,5,1,3,8)!

B. Fogalmazd meg általánosan, hogy milyen tulajdonságú szám lesz az algoritmus befejeződésekor a piramis tetején!

C. Fogalmazd meg általánosan, hogy milyen viszonyban van az algoritmus végrehajtása után az I sorszámú dobozban levő szám a fölötte és az alatta levőkkel (feltéve, hogy van szám fölötte, illetve alatta)!

Kilencedik-tizedik osztályosok

1. feladat: Válogatós (24 pont)

Lányok (N) és fiúk (M) ülnek egy sorban, vegyesen (N+M>0). Szeretnénk úgy ültetni őket, hogy az első N széken üljenek a lányok, a következő M széken pedig a fiúk.

A feladat megoldására a következő algoritmust találtuk ki:

1. Az első széken levőt felállítjuk (kezdetben ez lesz a szabad szék).

2. Az utolsó széktől visszafelé, de legfeljebb a szabad székig haladva keresünk egy lányt.

3. Ha találunk, akkor megjegyezzük, hogy legközelebb már csak innentől kezdve kell lányt keresni, és a lányt átültetjük a szabad székre.

4. Ezután az első széktől, de legfeljebb a szabad székig haladva keresünk egy fiút.

5. Ha találunk, akkor megjegyezzük, hogy legközelebb már csak innentől kezdve kell fiút keresni, és a fiút átültetjük a szabad székre.

6. Ha egyik keresés során sem értük el a szabad széket, akkor a 2. ponttól kezdve folytatjuk a keresést.

7. A szabadon maradt székre ültetjük azt, akit az első lépésben felállítottunk.

Példa:

`(Dóra,Endre,Ágnes,István)` ⇒ `(-,Endre,Ágnes,István)` ⇒ `(Ágnes,Endre,-,István)` ⇒ `(Ágnes,-,Endre,István)` ⇒ `(Ágnes,Dóra,Endre,István)`

A. Mi lesz az eredmény, ha kezdetben a székeken ülők sorrendje (azaz a kezdő ülésrend): `(Erika,János,Nóra)`? A fenti példához hasonlóan lépésenként írd le!

B. Mi lesz az eredmény, ha a kezdő ülésrend: `(Anna,Emese,Béla,Enikő,András,Jakab,Eszter,Móni,László,Piri,Péter)`? A fenti példához hasonlóan lépésenként írd le!

C. Van olyan kezdő ülésrend, hogy csak egyetlen gyereknek kell mozognia. Melyik gyereknek, milyen kezdő ülésrend esetén?

D. Van olyan kezdő ülésrend, hogy minden gyereknek mozognia kell. Melyik ez a kezdő ülésrend?

2. feladat: Szakaszrajzolás (12 pont)

Milyen hibák vannak az alábbi szakaszrajzoló eljárásban, amely az (x1,y1) és az (x2,y2) pontok között próbál meg szakaszt rajzolni? (A koordináták egész számok. Feltehetjük, hogy a végpontok

biztosan a képernyőn vannak.) Hogyan jellemezhetők azok a szakaszok, amelyeket ez a szakaszrajzoló hibásan rajzol fel?

```
Szakasz(x1,y1,x2,y2):
  dy:=(y2-y1)/(x2-x1); py:=y1; px:=x1
  Ciklus amíg px<x2
    rajzol(px,py) {Kirajzolja az adott pontot}
    py:=py+dy; px:=px+1
  Ciklus vége
Eljárás vége.
```

3. feladat: Rekurzió (23 pont)

Sorozatokon az alábbi függvényeket értelmezzük:

`üres(`*`sorozat`*`)`	igaz, ha a *sorozat* üres, hamis, ha nem
`első(`*`sorozat`*`)`	a *sorozat* első eleme
`utolsó(`*`sorozat`*`)`	a *sorozat* utolsó eleme
`elsőnélküli(`*`sorozat`*`)`	a *sorozat* első elem nélküli (rész)sorozata
`utolsónélküli(`*`sorozat`*`)`	a *sorozat* utolsó elem nélküli (rész)sorozata
`elsőnek(`*`elem`*`,`*`sorozat`*`)`	az új *elem*mel az elején bővített *sorozat*
`utolsónak(`*`elem`*`,`*`sorozat`*`)`	az új *elem*mel a végén bővített *sorozat*

Az alábbi rekurzív (azaz önmagukat hívó) függvényeket karaktersorozatokra (sztringekre) alkalmazzuk:

```
Egyik(S):
  Ha üres(S) vagy üres(elsőnélküli(S)) akkor Egyik:=S
  különben Egyik:=utolsónak(első(S),elsőnek(utolsó(S),
                   Egyik(elsőnélküli(utolsónélküli(S)))))
Függvény vége.
```

```
Másik(S):
  Ha üres(S) vagy üres(elsőnélküli(S)) akkor Másik:=S
  különben Másik:=utolsónak(első(S),
                   elsőnek(első(elsőnélküli(S)),
                   Másik(elsőnélküli(elsőnélküli(S)))))
Függvény vége.
```

A) Mi az eredménye az `Egyik("ABCDEF")` függvényhívásnak? Fogalmazd meg általánosan is!

B) Mi az eredménye a `Másik("ABCDEF")` függvényhívásnak? Fogalmazd meg általánosan is!

C) Mi az eredménye az `Egyik("ABCDE")` függvényhívásnak?

D) Mi az eredménye a `Másik("ABCDE")` függvényhívásnak?

4. feladat: Mit csinál? (20 pont)

A következő algoritmus az A vektorban található nemnegatív egész számok alapján állítja elő a C vektort.

```
Mitcsinál(A,C):
  Ciklus i=1-től N-ig
    B(i):=1; C(i):=-1
    Ciklus j=1-től N-ig
      Ha A(i)>A(j) akkor B(i):=B(i)+1
    Ciklus vége
  Ciklus vége              {*}
  Ciklus i=1-től N-ig
    C(B(i)):=A(i)
  Ciklus vége              {**}
  Ciklus i=2-től N-ig
    Ha C(i)<C(i-1) akkor C(i):=C(i-1)
  Ciklus vége              {***}
Eljárás vége.
```

A. Mit tartalmaz a B vektor a {*} jellel jelölt ciklus lefutása után?

B. Mit tartalmaz a C vektor a {**} jellel jelölt ciklus lefutása után?

C. Mit tartalmaz a C vektor a {***} jellel jelölt ciklus lefutása után?

D. Mit csinál az algoritmus?

5. feladat: Kódolás (22 pont)

Egy tömörítő eljárás a tömörítendő jelsorozat azonos elemekből álló részsorozatait három karakterrel kódolja: a ⊕-jellel, a sorozat hosszából mint ascii-kódból előállított karakterrel és magával az ismétlődő karakterrel. Az eljárásnak az X szöveges állomány a bemenete és a Z szöveges állomány a kimenete.

```
Eljárás Tömörítés:
  Nyit(X); Nyit(Z); Olvas(X,k)
  Ciklus amíg nem filevége(X)
    Csoportolvasás(X,db,kar,k); Ír(Z,⊕+Karakter(db)+kar)
  Ciklus vége
  Zár(X); Zár(Z);
Eljárás vége.

Eljárás Csoportolvasás(X,db,kar,k):
  kar:=k; db:=1; Olvas(X,k)
  Ciklus amíg k=kar
    db:=db+1; Olvas(X,k)
  Ciklus vége
Eljárás vége.
```

A. Milyen hibák vannak az eljárásban?

B. Hogyan lehet ezeket kijavítani? (Ne új eljárást írj!)

C. Milyen esetekben lesz hosszabb a tömörített állomány, mint a tömörítetlen?

D. Mit kell módosítani az eljárásban, hogy a tömörített állomány semmilyen esetben ne lehessen hosszabb az eredetinél?

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rendezés (20 pont)

Az alábbi eljárások az N db különböző egész számot tartalmazó T tömb növekvő sorrendbe rendezésére készültek.

A. Mi a `B` változó szerepe az `Egyik`, illetve a `B` és a `D` változó szerepe a `Másik` eljárásban?

B. Hányszor hajtja végre `Egyik` a {*}, illetve `Másik` a {*}, {**}, {***} jellel megjelölt sorokat (az összehasonlító műveleteket)?

C. Melyik eljárásban kevesebb az összehasonlítások száma?

```
Egyik(T,N):
  Ciklus i=0-tól N-1-ig
    A:=T(i); B:=i
    Ciklus j=i+1-től N-1-ig
       Ha T(j)<A akkor A:=T(j); B:=j              {*}
    Ciklus vége
    T(B):=T(i); T(i):=A
  Ciklus vége
Eljárás vége.
```

```
Másik(T,N):
  Ciklus i=0-tól N/2-1-ig
    Ha T(i)<T(N-i-1)                                  {*}
        akkor  A:=T(i); B:=i; C:=T(N-i-1); D:=N-i-1
      különben A:=T(N-i-1); B:=N-i-1; C:=T(i); D:=i
    Ciklus j=i+1-től N-i-2-ig
      Ha T(j)<A akkor A:=T(j); B:=j                   {**}
      Ha C<T(j) akkor C:=T(j); D:=j                   {***}
    Ciklus vége
    T(B):=T(i); T(D):=T(N-i-1); T(i):=A; T(N-i-1):=C
  Ciklus vége
Eljárás vége.
```

2. feladat: Rekurzió (20 pont)

Sorozatokon az alábbi függvényeket értelmezzük:

`üres(`*`sorozat`*`)`	igaz, ha a *sorozat* üres, hamis, ha nem
`első(`*`sorozat`*`)`	a *sorozat* első eleme
`utolsó(`*`sorozat`*`)`	a *sorozat* utolsó eleme
`elsőnélküli(`*`sorozat`*`)`	a *sorozat* első elem nélküli (rész)sorozata
`utolsónélküli(`*`sorozat`*`)`	a *sorozat* utolsó elem nélküli (rész)sorozata
`elsőnek(`*`elem,sorozat`*`)`	az új *elem*mel az elején bővített *sorozat*
`utolsónak(`*`elem,sorozat`*`)`	az új *elem*mel a végén bővített *sorozat*

Az alábbi kölcsönösen rekurzív (azaz önmagukat és egymást hívó) függvénypárokat páros karakterszámú karaktersorozatokra (sztringekre) alkalmazzuk.

```
Egyik(S):
  Ha üres(S) vagy üres(elsőnélküli(S)) akkor Egyik:=S
  különben Egyik:=utolsónak(első(S),elsőnek(utolsó(S),
                  Másik(elsőnélküli(utolsónélküli(S)))))
Függvény vége.
```

```
Másik(S):
  Ha üres(S) vagy üres(elsőnélküli(S)) akkor Másik:=S
  különben Másik:=elsőnek(első(S),utolsónak(utolsó(S),
                  Egyik(elsőnélküli(utolsónélküli(S)))))
Függvény vége.
```

```
Alfa(S):
  Ha üres(S) vagy üres(elsőnélküli(S)) akkor Alfa:=S
  különben Alfa:=utolsónak(első(S),
                elsőnek(első(elsőnélküli(S)),
                Béta(elsőnélküli(elsőnélküli(S)))))
Függvény vége.

Béta(S):
  Ha üres(S) vagy üres(elsőnélküli(S)) akkor Béta:=S
  különben Béta:=elsőnek(utolsó(S),
                utolsónak(utolsó(utolsónélküli(S)),
                Alfa(utolsónélküli(utolsónélküli(S)))))
Függvény vége.
```

A. Mi az eredménye az `Egyik("ABCDEF")` függvényhívásnak? Fogalmazd meg általánosan is!

B. Mi az eredménye a `Másik("ABCDEF"")` függvényhívásnak? Fogalmazd meg általánosan is!

C. Mi az eredménye az `Alfa("ABCDEFGHIJ")` függvényhívásnak? Fogalmazd meg általánosan is!

D. Mi az eredménye a `Béta("ABCDEFGHIJ")` függvényhívásnak? Fogalmazd meg általánosan is!

3. feladat: Szövegkeresés (18 pont)

Az alábbi eljárás egy szövegben (`s`) keres egy másik szöveget (`sminta`). A szöveg csak az angol ábécé 26 nagybetűjét tartalmazza. A `kód` függvény minden betűhöz egy 0 és 25 közötti számot rendel (különböző betűkhöz különböző számot).

```
Eljárás Keresés(s,sminta,siker,i):
  Konstans q: Egész(33554393)
           d: Egész(26)
  m:=Hossz(sminta); dh:=1; s1:=0; s2:=0
  Ciklus i=1-től m-1-ig
    dh:=(d*dh) mod q
  Ciklus vége
  Ciklus i=1-től m-ig
    s1:=(s1*d+Kód(sminta(i))) mod q
  Ciklus vége
  Ciklus i=1-től m-ig
    s2:=(s2*d+Kód(s(i))) mod q
  Ciklus vége
  i:=1
  Ciklus amíg (s1≠s2) és i≤Hossz(s)-m
    s2:=(s2+d*q-Kód(s(i))*dh) mod q
    s2:=(s2*d+Kód(s(i+m))) mod q; i:=i+1
  Ciklus vége
  siker:=(s1=s2)
Eljárás vége.
```

A. Milyen érték van a `dh` változóban? Mi az oka, hogy a maradékszámítást nem a ciklus lefutása után végezzük el?

B. Mi lesz `s1` és `s2` értéke az utolsó ciklus megkezdése előtt (mi közük az `s` és az `sminta` szöveges változókhoz)?

C. Hogyan, milyen elven változtatja `s2` értékét az utolsó ciklus?

D. Van olyan eset, amikor az algoritmus leáll, a `siker` változó igaz értékű, pedig az `s` változóban nem található meg az `sminta`. Mi ennek a feltétele?

4. feladat: Adatbázis (22 pont)

A SZÉTSZÓRT iskola több város több épületében működik. Az osztályokról és az osztályfőnökökről nyilvántartást készítettünk (lásd a két táblázatot). Városonként egy osztályfőnököt vezetőnek neveztek ki.

Osztályfőnökök				
azon	Név	szak	vezető	osztály
101	Nagy Péter	Matematika	0	1A
102	Kiss Anna	Matematika	104	1B
103	Kovács Pál	Magyar	101	2A
104	Németh Erika	Történelem	0	2B
105	Szabó János	Magyar	101	3A
106	Horváth Zsuzsanna	Kémia	104	3B
107	Kiss Etelka	Rajz	0	4C

Osztályok		
kód	Város	létszám
1A	Budapest	40
2A	Budapest	36
3A	Budapest	32
1B	Miskolc	24
2B	Miskolc	26
3B	Miskolc	26
4C	Debrecen	20

A táblázatok lekérdezésére használható a SELECT utasítás:

```
SELECT oszlopnév, oszlopnév, ...
      FROM táblázatnév, táblázatnév, ...
      WHERE logikai kifejezés;
```

A logikai kifejezésben műveleti jel (pl. `VAGY`, `ÉS`, =, <, >=), állandó (pl. `"Miskolc"`), oszlopnév (pl. `város`) vagy zárójelben egy újabb, beágyazott SELECT utasítás használható.

A lekérdezés azokat a sorokat (két táblázat esetén: sorpárokat) választja ki a táblázat(ok)ból, amelyekre a logikai kifejezés teljesül. Minden kiválasztott sorból, illetve sorpárból csak a megnevezett oszlopok adatait kapjuk eredményül. A nem beágyazott SELECT utasítás kiírja az eredményt a képernyőre. Például

```
SELECT kód FROM osztályok WHERE város="Miskolc" OR város="Debrecen";
```

kiírja a miskolci és a debreceni osztályok kódját.

A: Milyen parancs írja `ki`

A1: a 30-nál kisebb létszámú osztályok kódját és létszámát?

A2: a Debrecenben dolgozók nevét és osztályuk létszámát?

B: Mit ír ki az alábbi SELECT parancs?

```
SELECT név
  FROM Osztályfőnökök
  WHERE vezető = (SELECT azon
                  FROM Osztályfőnökök
                  WHERE név = "Nagy Péter");
```

C: Milyen parancs írja ki azon főnökök nevét, akiknek van Magyar szakos beosztottja?

5. feladat: Prioritási sor (20 pont)

A prioritási sor olyan adatszerkezet, amelybe az elemek a prioritásuknak (fontosságuknak) megfelelő helyre lépnek be, és amelyből a legmagasabb prioritású elem lép ki. Az alábbi eljárásokban a prioritási sort egy `hossz` darab elemet tartalmazó `K` tömb valósítja meg (kezdetben `hossz=0`).

```
Eljárás Sorba(e: ElemTip, p: Prioritás):
  hossz:=hossz+1
  K(hossz).elem:=e; K(hossz).pr:=p
  gy:=hossz; sz:=gy DIV 2
  Ciklus amíg sz≥1 és K(gy).pr>K(sz).pr
    Csere(K(gy),K(sz)); gy:=sz; sz:=gy DIV 2
  Ciklus vége
Eljárás vége.

Eljárás Sorból(e: ElemTip, p: Prioritás):
  e:=K(1).elem; p:=K(1).pr
  sz:=1; K(1):=K(hossz); hossz:=hossz-1
  gy:=2; Ha 2<hossz és K(2).pr<K(3).pr akkor gy:=3
  Ciklus amíg sz≤hossz DIV 2 és K(gy).pr>K(sz).pr
    Csere(K(sz),K(gy)); sz:=gy; gy:=sz*2
    Ha gy<hossz és K(gy).pr<K(gy+1).pr akkor gy:=gy+1
  Ciklus vége
Eljárás vége.
```

A. Hol lesz a tömbben a legmagasabb prioritású elem?

B. Milyen összefüggés van a tömbben az egyes elemek sorrendje és prioritása között?

C. Mi a ciklusok maximális lépésszáma a két eljárásban?

D. Mi lesz lépésenként a K tömbben az alábbi műveletsor hatására: `Sorba(A,5); Sorba(B,3); Sorba(C,7); Sorba(D,2); Sorba(E,4); Sorból(X,Y)`?

2002. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mókus (23 pont)

Egy patakban N követ raktak le átjárónak. A köveken M mókus ugrál át a túlsó partra úgy, hogy mindegyik csak bizonyos kövekre (pl. a parttól számítva minden második, minden negyedik kőre) ugrik.

Készíts programot, amely beolvassa a kövek számát (1≤N≤20), a mókusok számát (1≤M≤20) és azt, hogy az egyes mókusok minden hányadik kőre ugranak (1≤K(i)≤N, ahol 1≤i≤M), majd megadja, hogy mely kövekre nem lép egyetlen mókus sem (0 legyen a válasz, ha nincs ilyen kő), illetve hogy melyekre lép a legtöbb!

Példa:

```
Bemenet: N=15, M=3, K=(2,3,4)
Kimenet: Egy mókus sem lép ezekre: 1,5,7,11,13
         A legtöbb mókus lép ezekre: 12
```

2. feladat: Torlódás (25 pont)

A szláv nyelvekben gyakori a mássalhangzó-torlódás (egyik szép példája a cseh *zmrzlina* – fagylalt – szó), de a magyar nyelvben is előfordul, hogy egyes szavakban egymás mellé kerül több mássalhangzó (pl. *parancsnok,* amelyben három mássalhangzó – n, cs és n – áll egymás mellett).

Készíts programot, amely beolvas egy maximum 255 (ékezetes és ékezet nélküli) kisbetűből álló szót, majd megadja, hogy hányadik karaktertől kezdődik benne a legnagyobb elemszámú mássalhangzó-torlódás, és az hány mássalhangzóból áll! Ha több egyforma hosszú torlódás lenne benne, akkor közülük csak az elsőt kell megadni!

Az egy- és többjegyű mássalhangzók rövid változatát (pl. *k, t, cs, dz, ty, dzs)* a program egyetlen mássalhangzónak, hosszú változatukat (pl. *kk, tt, ccs, ddz, tty, ddzs)* két mássalhangzónak vegye. (Például az *attrakció* szóban a leghosszabb torlódás a második betűnél kezdődik és három a hossza, a *szennyes* szóban pedig a negyedik karakternél kezdődik és kettő a hossza) Egyes szavakról (pl. *házsor)* a jelentésük ismerete nélkül nem állapítható meg, hogy többjegyű mássalhangzó vagy több egymás mellett álló mássalhangzó van-e bennük. Minden ilyen esetben a program vegye úgy, hogy a mássalhangzó többjegyű (azaz pl. a *házsor* szóban a kétjegyű *zs*-vel, mint egyetlen mássalhangzóval kell számolni.)

Példák:

```
fapipa              1 1
zmrzlina            1 5
parancsnok          5 3
attrakció           2 3
szennyes            4 2
házsor              1 1
helyzetünkben       9 3
```

3. feladat: „Egyes” számok (27 pont)

Adjuk össze egy legfeljebb 100 jegyű természetes szám számjegyeit! Ha a kapott szám nem egyjegyű, akkor ennek újra adjuk össze a számjegyeit, s az eljárást folytassuk, amíg egyjegyű számot nem kapunk. Egyes kiinduló számok esetén a végeredmény 1 lesz. Nevezzük ezeket a kiinduló számokat „egyes” számoknak.

`91: (91→10→1)  1998: (1998→27→9)  1999: (1999→28→10→1)`

Készíts programot, amely beolvas egy legfeljebb 100 jegyű természetes számot, majd kiírja, hogy a szám „egyes” szám-e!

Példa:

```
Bemenet:91      Kimenet: EGYES
Bemenet:1998    Kimenet: NEM EGYES
Bemenet:1999    Kimenet: EGYES
```

Kilencedik-tizedik osztályosok

1. feladat: Tipp (15 pont)

Egy szerencsejátékban a résztvevők sorban egymás után egy-egy 1 és M közötti számot tippelnek. Az nyer, aki elsőként tippelt arra a számra, amelyre a résztvevők közül a legtöbben tippeltek. Ha több ilyen szám is van, akkor az összes ilyen szám első tippelője nyer.

Írj programot, amely fogadja és feldolgozza a tippeket, majd megadja a nyertes játékos sorszámát, a nyertes számot, és azt, hogy hányan választották ezt a számot!

A TIPP.BE állomány első sorában a versenyzők száma ($1 \le N \le 5000$) és a tipp maximális értéke ($1 \le M \le 1\ 000\ 000$) van, egyetlen szóközzel elválasztva. A következő N sorban van az egyes versenyzők tippje.

A TIPP.KI állományba annyi sort kell írni, ahány győztes van (tippjük szerint növekvő sorrendben)! Minden sorban három szám legyen: a győztes sorszáma, a tippje, valamint az, hogy hányan tippelték ezt a számot!

Példa:

```
TIPP.BE             TIPP.KI
6 100               3 15 2
25                  2 20 2
20
15
15
30
20
```

2. feladat: Ütemezés (20 pont)

Mekk Elek ezermester népszerű vállalkozó, sokan keresik fel megrendelésekkel. Minden munkája pontosan egy napig tart. Minden megrendelés határidős, és amit elvállal, határidőre el is végzi. A mester a következő évre beérkezett megrendelések közül a lehető legtöbbet akarja elvállalni, de egyszerre csak egy munkán tud dolgozni.

Írj programot a következő évi megrendelések egy lehető legnagyobb elemszámú részhalmazának a kiválasztására és ütemezésére annak érdekében, hogy a mester a lehető legtöbb munkát határidőre el tudjon végezni! A programnak egy ilyen ütemezést kell eredményül adnia!

Az UTEMEZ.BE állomány első sora a megrendelések számát ($1 \le N \le 10\,000$) tartalmazza. A következő N sor mindegyikében egy-egy H pozitív egész szám, az adott megrendelés határideje áll ($1 \le H \le 365$).

Az UTEMEZ.KI állomány első sorában a kiválasztott munkák M száma legyen! A következő M sor mindegyikébe két számot kell írni! Az első szám a kiválasztott munka száma legyen, a második pedig annak a napnak a sorszáma, amelyiken az adott munkát el kell végezni! Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni az állományba!

Példa:

```
UTEMEZ.BE           UTEMEZ.KI
6                   5
3                   5 1
2                   1 3
7                   2 2
4                   4 4
2                   3 7
1
```

Megjegyzés: `5 1` helyett pl. a `6 1`, `3 7` helyett a `3 5` vagy a `3 6` válasz is jó.

3. feladat: Üvegválogatás (20 pont)

Egy palackozó üzembe N db ládában érkeznek be az üvegek. Alakjuk szerint K fajta üveget különböztetnek meg. Ismert, hogy az egyes ládákban hány darab üveg van az egyes fajtákból. A palackozáshoz az üvegeket a fajtájuk szerint szét kell válogatni. Minden üvegfajta számára kijelölnek egy ládát (a meglévő N közül), és a többi ládából az adott fajta üveget ebbe a ládába rakják át. A cél az, hogy a lehető legkevesebb üveget kelljen átrakni a válogatás során.

Írj programot, amely kiszámítja, hogy legkevesebb hány üveget kell átrakni, és ez mely ládák kijelölésével érhető el!

A VALOGAT.BE állomány első sorában a ládák ($2 \le N \le 10$) és a fajták ($2 \le K \le N$) száma van. A következő N sor mindegyike egy-egy láda tartalmát írja le. Minden sor pontosan K db nemnegatív

egész számot tartalmaz, ahol a J-edik szám a ládában található J-edik üvegfajta darabszáma (1≤J≤K). (A ládák elég nagyok ahhoz, hogy mindegyikbe tetszőleges számú üveg beleférjen.)

A VALOGAT.KI állományba két sort kell írni! Az első sorban a válogatáshoz minimálisan szükséges átrakások száma legyen! A második sor pontosan K számot tartalmazzon, ahol a J-edik szám annak a ládának a sorszáma legyen, amelyiket a J-edik üvegfajta számára kijelöltünk! Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni!

Példa:

VALOGAT.BE	VALOGAT.KI
5 4	5 8
1 7 2 6	5 1 3 4
3 1 2 4	
3 1 5 6	
6 4 7 8	
6 7 1 4	

4. feladat: Tükörszó (20 pont)

Egy szót tükörszónak nevezünk, ha balról és jobbról kiolvasva betűről betűre megegyezik. (Tehát minden egybetűs szó tükörszó.) Minden szó felbontható részekre úgy, hogy minden rész tükörszó legyen. Minimálisnak nevezzük az olyan felbontást, amely egy szót a lehető legkevesebb tükörszóra szed szét.

Írj programot, amely kiszámítja, hogy egy adott szó minimális felbontása hány tükörszóból áll!

A TUKOR.BE állomány egyetlen sorában egy legfeljebb 100 karakterből álló ***S*** szó van.

A TUKOR.KI állományba egyetlen számot kell írni: az ***S*** szó minimális felbontásához szükséges tükörszavak számát!

Példa:

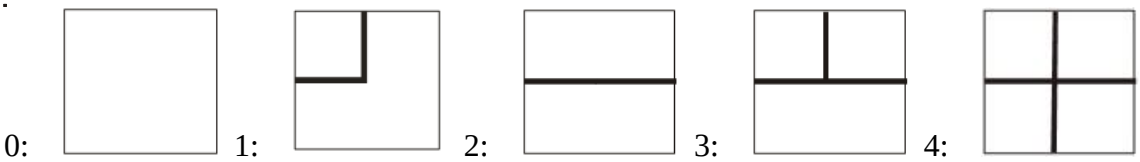
TUKOR.BE	TUKOR.KI
abbakabadara	5

Megjegyzés: a megoldás értéke az `abba k aba d ara` felbontáshoz tartozik.

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Villamos (20 pont)

Egy négyzetrács-szerkezetű városban különleges villamosok közlekednek, ugyanis olyan pályán járnak, amelynek a négyzet alakú elemeit el tudják forgatni. Az elemek a következők:



Minden elem négyféle helyzetben állhat:

0: az ábrán látható módon

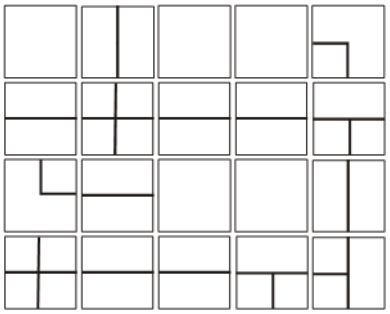
1: 90 fokkal jobbra elforgatva

2: 180 fokkal jobbra elforgatva

3: 270 fokkal jobbra elforgatva

Írj programot, amely megadja, hogy a villamos egy adott helyről egy másikra minimálisan hány lépésben (azaz a kezdőhelyet nem számítva hány elem érintésével) juthat el, illetve minimálisan hány lépésben juthat el akkor, ha azt az elemet, amelyen éppen áll, el tudja forgatni jobbra 90 fokkal!

(Figyelem: a forgatás is lépésnek számít. Ugyanaz az elem több lépésben többször egymás után is elforgatható jobbra 90 fokkal.)



A VILLAMOS.BE állomány első sorában a négyzetrács sorainak ($1 \le N \le 100$) és oszlopainak ($1 \le M \le 100$) a száma van. A következő N sorban soronként M számjegy-pár (két szorosan egymás mellé írt számjegy) található egy-egy szóközzel elválasztva; mindegyik sor a négyzetrács egy sorát írja le. A négyzetrács minden elemét a fenti ábrán megadott azonosító számból és az elforgatás kódjából álló számjegy-párral adjuk meg.

A bemenő állomány utolsó sorában négy egész szám van: a kezdőhely sor- és oszlopindexe, valamint a célhely sor- és oszlopindexe.

A VILLAMOS.KI állomány első sorába azt a minimális lépésszámot kell írni, amely elegendő ahhoz, hogy a villamos eljusson a kezdőhelyről a célhelyre; a második sorba pedig ugyanezt a számot abban az esetben, ha a villamos elforgathatja azt az elemet, amelyen éppen áll vagy áthalad! A lépésszám legyen –1, ha nem lehet eljutni a kezdőhelyről a célhelyre!

Példa:

VILLAMOS.BE	VILLAMOS.KI
4 5	10
00 21 00 00 13	6
20 40 20 20 32	
11 20 00 00 21	
40 20 20 32 33	
1 2 4 1	

Megjegyzés:

Út az 1. esetben: (1,2),(2,2),(2,3),(2,4),(2,5),(3,5),(4,5),(4,4),(4,3),(4,2),(4,1)

Út a 2. esetben: (1,2),(2,2),(2,1),fordít,(3,1),fordít,(4,1)

2. feladat: Mozgat (20 pont)

Minden szövegszerkesztővel végezhető kivágás-beszúrás művelet. Minden műveletet egy A, B, C számhármas ír le, ami azt jelenti, hogy a szöveg A-tól B-ig terjedő sorait (A-t és B-t is beleértve) kivágjuk, és beszúrjuk a C-edik sor mögé. (Az A, B és C sorszámok a művelet elvégzése előtt értendők.)

Egy N sorból álló szövegre K-szor alkalmazunk kivágás-beszúrás műveletet.

Írj programot, amely kiszámítja, hogy

- a szöveg első 10 sora hova került a műveletek elvégzése után;
- az eredeti szöveg mely sorai kerültek az első 10 sorba a műveletek hatására!

A MOZGAT.BE állomány első sora két egész számot tartalmaz: az első a szöveg sorainak a száma ($10 \le N \le 1\,000\,000$), a második pedig a műveletek száma ($1 \le K \le 1000$). A további K sor mindegyikében három egész szám van: A, B és C, amelyek egy-egy műveletet írnak le. A számokra teljesülnek a következő egyenlőtlenségek: $1 \le A \le B \le N$, továbbá $0 \le C < A$ vagy $B \le C \le N$. Ha C=0, akkor a kivágott szöveget az első sor elé kell beszúrni.

A MOZGAT.KI állomány két sorába 10-10 számot kell írni! Először azoknak a szövegsoroknak a sorszámát kell felsorolni, ahová az eredeti szöveg első 10 sora került a műveletek hatására! Azután az eredeti szöveg azon sorainak a sorszámát kell felsorolni, amelyek a műveletek hatására az első 10 sorba kerültek át!

Példa:

```
MOZGAT.BE            MOZGAT.KI
1000 4               805 2 3 806 807 808 809 810 115 1
1 9 10               10 2 3 390 391 392 393 394 395 396
3 4 1
300 500 9
100 900 3
```

3. feladat: Ütemezés (15 pont)

Mekk Elek ezermester népszerű vállalkozó, sokan keresik fel megrendelésekkel. Minden megrendelt munkának ismeri a kezdési és a befejezési idejét. A mester a következő évre szóló megrendelések közül a lehető legtöbbet akarja elvállalni, de egyszerre csak egy munkán tud dolgozni.

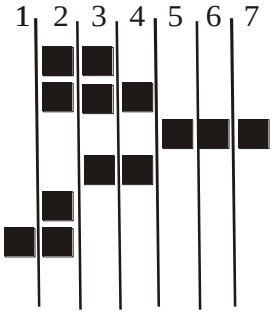
Írj programot, amely meghatározza a munkák egy lehető legnagyobb elemszámú részhalmazát úgy, hogy az összes kiválasztott munka elvégezhető legyen!

Az UTEMEZ.BE állomány első sora a megrendelések számát ($1 \le N \le 10\,000$) tartalmazza. A következő N sor mindegyike két pozitív egész számot tartalmaz, a megrendelt munka kezdési, illetve befejezési idejét ($1 \le K \le B \le 365$).

Az UTEMEZ.KI állomány első sorában a kiválasztott munkák M száma legyen! A második sorba M számot, a kiválasztott munkák sorszámát kell írni, tetszőleges sorrendben! Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni!

Példa:

```
UTEMEZ.BE            UTEMEZ.KI
6                    3
2 3                  6 3 4
2 4
5 7
3 4
2 2
1 2
```



4. feladat: Tükörszó (20 pont)

Egy szót tükörszónak nevezünk, ha balról és jobbról kiolvasva betűről betűre megegyezik. (Tehát minden egybetűs szó tükörszó.) Minden szóban található tükörszó, amin azt értjük, hogy ha kitörlünk belőle betűket, akkor tükörszót kapunk.

Írj programot, amely meghatározza egy adott szóban található leghosszabb tükörszó hosszát!

A TUKOR.BE állomány egyetlen sorában egy legfeljebb 100 karakterből álló ***S*** szó van.

A TUKOR.KI állományba egyetlen számot kell írni: az ***S*** szóban található leghosszabb tükörszó hosszát!

Példa:

```
TUKOR.BE             TUKOR.KI
abbakabadara         7
```

Megjegyzés: a megoldás értéke az **a**bb**a**k**aba**d**a**r**a** vastagon szedett részhez tartozik.

2002. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Betét (17 pont)

Az OKB (Országos Kuporgató Bank) új betétformát vezetett be. Kerek százas összeget lehet betenni a bankba, melyre havonta S% (1≤S≤10) kamatot fizetnek. Ezer forint felett még P% (0≤P≤S) prémium is jár. A kerek százasokon felül maradó részt a bank havonta kifizeti, amit otthon őrzünk. Ha az otthon őrzött pénz eléri vagy meghaladja a 100 forintot, akkor abból 100-at beteszünk a bankba.

Írj programot, amely megadja, hogy X betett összeg (100≤X≤100 000, X osztható 100-zal) esetén H (1≤H≤48) hónapon keresztül mennyi pénzünk lesz a bankban és mennyi lesz otthon!

Példák:

```
X=100, S=10, P=0, H=11
Hónap:     0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11
Bankban: 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 200, 200
Otthon:    0,  10,  20,  30,  40,  50,  60,  70,  80,  90,   0,  20
X=800, S=10, P=5, H=5
Hónap:     0,   1,   2,    3,    4,    5
Bankban: 800, 800, 900, 1000, 1200, 1300
Otthon:    0,  80,  60,   50,    0,   80
```

2. feladat: Szomszéd (28 pont)

Táblajátékokban gyakori, hogy bábuk egyes helyzetekben attól függően léphetnek, hogy a szomszédságukban milyen bábuk vannak.

Ehhez meg kell határozni az egyes mezők szomszédjainak a koordinátáit. A szomszédos mezők érintkezhetnek az oldalukkal, illetve a sarkukkal. A tábla szélén levő mezőknek lehet, hogy csak egyik irányban van szomszédjuk, de vehetjük úgy is, hogy a szomszédos mezők a tábla túlsó szélén vannak. A szomszédságot lehet csak a közvetlen szomszédokra értelmezni (1 távolságú szomszédok), s lehet nagyobb távolságra is.

Készíts programot, amely beolvassa a 100x100-as tábla egy mezőjének koordinátáit, majd az alábbi négy módszerrel kiírja a legfeljebb T távolságra levő szomszédjainak a koordinátáit! A bal felső mező koordinátája (1,1).

A. Egy hely közvetlen szomszédjai azok a mezők, amelyek oldalukkal vagy sarkukkal érintkeznek az adott hellyel.

2	2	2	2	2	
2	1	1	1	2	
2	1	X	1	2	
2	1	1	1	2	
2	2	2	2	2	

Szomszédok max. 2 egység távolságra

Szomszédok 1 egység távolságra

X	1				
1	1				

B. Egy hely közvetlen szomszédjai azok a mezők, amelyek oldalukkal érintkeznek az adott hellyel.

		2			
	2	1	2		
2	1	X	1	2	
	2	1	2		
		2			

Szomszédok max. 2 egység távolságra

Szomszédok 1 egység távolságra

1					
X	1				
1					

C. Egy hely közvetlen szomszédjai azok a mezők, amelyek oldalukkal vagy sarkukkal érintkeznek az adott hellyel. A szélen levő mezők egyes szomszédjai a túloldalon vannak.

2	2	2	2		2
1	1	1	2		2
1	X	1	2		2
1	1	1	2		2
2	2	2	2		2

Szomszédok max. 2 egység távolságra

D. Egy hely közvetlen szomszédjai azok a mezők, amelyek oldalukkal érintkeznek az adott hellyel. A szélen levő mezők egyes szomszédjai a túloldalon vannak.

2	1	X	1	2	
	2	1	2		
		2			
		2			
	2	1	2		

Szomszédok max. 2 egység távolságra

3. feladat: DNS (30 pont)

A DNS mesterséges előállításáért folytatott kísérletek során sikerült előállítani egy óriásmolekula-szálat. Ennek leírása legfeljebb 255 karakterből áll (C, G, A és T betű lehet benne). Ismétlődésnek nevezünk egy legalább 2 karakterből álló sorozatot, ha a DNS-leírásban legalább kétszer előfordul (egymást nem átfedően).

Írj programot, amely beolvassa a DNS-leírást, majd megadja a benne szereplő leghosszabb ismétlődő szakaszt (ha van olyan), valamint a legtöbbször ismétlődő, legalább 3 karaktert tartalmazó szakaszt és ismétlődései számát!

Ha valamelyik részfeladat nem oldható meg, akkor a NINCS ILYEN ISMÉTLŐDÉS szöveget kell az eredmény helyett kiírni! Ha valamelyik részfeladatra több megoldás is van, akkor bármelyik megadható eredménynek!

Példák:

```
DNS: CGACCGACCGAT
Leghosszabb ismétlődő: CGAC
Legtöbbször ismétlődő: CGA, 3 ismétlődés
DNS: ACGTCG
Leghosszabb ismétlődő: CG
Legtöbbször ismétlődő: NINCS ILYEN ISMÉTLŐDÉS
```

Kilencedik-tizedik osztályosok

1. feladat: Épít (20 pont)

Egy építkezés befejezéséhez N különböző munkát kell elvégezni, a munkákat az 1,…,N számokkal azonosítjuk. Minden munka pontosan egy nap alatt teljesíthető, egy napon több munka is végezhető. A terv alapján azonban tudjuk, hogy bizonyos munkát előbb kell elvégezni, mint másokat. Pontosabban, a terv (A, B) párok halmazát tartalmazza, ami azt jelenti, hogy az A munkát előbb kell elvégezni, mint a B munkát.

Írj programot, amely

A. kiszámítja, hogy legkevesebb hány nap kell az összes munka elvégzéséhez;

B. és megadja a munkáknak egy olyan beosztását, amely teljesíti a követelményeket!

A `EPIT.BE` állomány első sora két egész számot tartalmaz: az elvégzendő munkák számát ($1 \le N \le 200$) és a tervben megadott megelőzési párok számát ($1 \le T \le 10\,000$). A további T sor mindegyikében két egész szám van ($1 \le A \le N$, $1 \le B \le N$), ami azt jelenti, hogy az A munkát előbb kell elvégezni, mint B-t.

A `EPIT.KI` állomány első sorába a lehető legkevesebb napok M számát kell írni, ami alatt az összes munkát el lehet végezni! Ezt követően pontosan M sornak kell lennie! Az állomány I+1-edik sora azon munkák sorszámát tartalmazza, amelyeket az I-edik napon kell elvégezni! Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni! Ha nem lehet a tervben megadott feltételeket teljesíteni, akkor az első sorba 0-t kell kiírni!

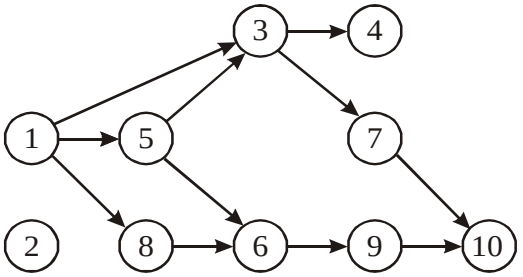
Példa:

`EPIT.BE`

```
10 11
1 3
3 4
1 5
5 3
3 7
5 6
1 8
8 6
6 9
9 10
7 10
```

`EPIT.KI`

```
5
1 2
5 8
3 6
4 7 9
10
```



2. feladat: Felvételi (20 pont)

Bergengócia egyetemeire N informatikus szakra jelentkezhetnek a felvételizők, a szakokat sorszámukkal azonosítjuk. Közös felvételi vizsgát tesznek, azaz mindenkinek egyetlen pontszáma van. Minden diák meghatározhatja a felvételi lapján, hogy melyik egyetemre akar kerülni, a szokásos módon prioritási sorrendben. Ezen kívül tudjuk, hogy melyik szakra hányan vehetők fel.

A felvételi ponthatárokat úgy kell megállapítani, hogy teljesüljenek az alábbi feltételek:

- 60 pont alatt senkit sem lehet felvenni.
- Minden jelentkező arra a szakra kerül be, amelyik az első a prioritási sorrendjében, amelyen a ponthatár nem nagyobb, mint az ő pontszáma.
- Minden szakra az így felvett jelentkezők száma nem haladja meg a szakra megadott keretszámot, kivéve azt az esetet, amikor egyel magasabb pontszámmal a szakra a keretszámnál kevesebb kerülne be, ekkor a felvehetők száma a keretszám 110%-a (lefelé kerekítve) lehet.

- Bármely két, azonos pontszámú jelentkező esetén, ha jelentkeztek ugyanarra a szakra, akkor vagy mindkettőt felvették, vagy egyiket sem vették fel.
- A ponthatár megállapítás olyan legyen, hogy összességében a lehető legtöbb jelentkezőt vegyenek fel.

Írj programot, ami meghatározza, hogy melyik szakon mennyi lesz a felvételi ponthatár (a felvett tanulók pontszámai közül a legkisebb, ha senki nem került be a szakra, akkor 0), és hogy melyik tanuló melyik szakra került be!

A `FELVET.BE` állomány első sorában a szakok ($1 \le N \le 100$) és a felvételizők ($1 \le M \le 9000$) száma van. A második sorban N szám van: minden szakra a maximálisan felvehető diákok száma, a keretszám ($0 \le K(i) \le 1000$). A következő M sorban a diákok adatai vannak: a pontszámuk ($0 \le P(j) \le 120$, $1 \le j \le 6$), majd azon szakok sorszáma a jelentkezés sorrendjében, ahova be szeretne kerülni.

A `FELVET.KI` állomány első sorába N db számot kell írni: az i. szám az i. szakon a felvételi ponthatár, illetve 0, ha egyetlen diákot sem vettek fel! A második sorba M db számot kell írni: az i. szám annak a szaknak a sorszáma, ahová az i. diák bekerült, illetve 0, ha a diákot nem vették fel sehová!

Példa:

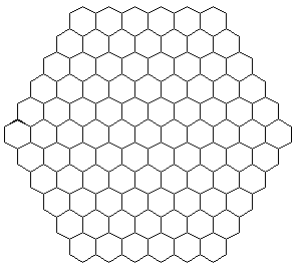
`FELVET.BE`

```
4 5
1 2 2 3
98 3 2 1 4
81 1 3 2
82 4
92 3 1
0 1 2 3 4
```

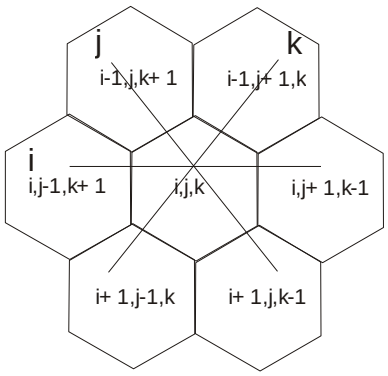
`FELVET.KI`

```
81 0 92 82
3 1 4 3 0
```

3. feladat: Hatszög (18 pont)



Egy szabályos hatszög alakú térképet szabályos hatszögekre bontottunk úgy, hogy minden oldalán N darab kisebb hatszög lett. A szemben levő csúcsokat összekötő átlókra az ábra szimmetrikus, így a hatszögeket 3 index-szel azonosíthatjuk (mindegyik az egyik tengely mentén állandó). Az **`i`** index a vízszintes tengely mentén állandó, felfelé csökken, lefelé nő. A **`j`** index a jobbra lefelé haladó tengely mentén állandó, ettől balra csökken, jobbra pedig nő. A **`k`** index pedig a balra lefele haladó tengely mentén állandó, ettől balra növekszik, jobbra pedig csökken.



Így az (i,j,k) indexű elem szomszédai indexei a jobboldali ábra szerint alakulnak. Legyen a középső kis hatszög indexe a (0,0,0)! Minden egyes pontnak ismerjük a tengerszint feletti magasságát.

Készíts programot, amely a (`p,q,r`) indexű pontból meghatározza a legrövidebb olyan ún. vízszintes út hosszát az (`x,y,z`) pontba, melynek során a tengerszint feletti magasság nem változik!

A `HATSZOG.BE` állomány első sorában a nagy hatszög mérete ($1 \le N \le 80$) van. A következő 2*N-1 sorban annyi tengerszint feletti magasság van ($0 < \text{magasság} \le 1000$), amennyi a térkép egy-egy sorához szükséges. Az utolsó sorban a kezdő- (p,q,r) és a végpozíció (x,y,z) indexei találhatók, egy-egy szóközzel elválasztva.

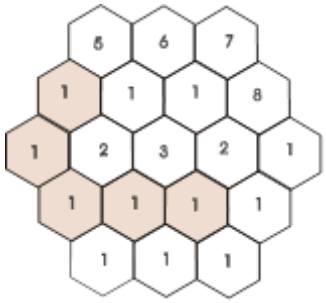
A HATSZOG.KI egyetlen sorába a legrövidebb (p,q,r)-ből (x,y,z)-be vezető vízszintes út hosszát kell írni! Ha ilyen út nincs, akkor a kiírt szám legyen -1!

Példa:

HATSZOG.BE
```
3
5 6 7
1 1 1 8
1 2 3 2 1
1 1 1 1
1 1 1
-1 -1 2 1 0 -1
```

HATSZOG.KI
```
4
```

Az út kiszínezve:



4. feladat: Lefed (17 pont)

Adott a síkon M pont. El kell helyezni a koordináta-rendszer tengelyeivel párhuzamos egyeneseket, úgy, hogy minden megadott pont rajta legyen legalább egy egyenesen. Azonban, az egyeneseket csak párosával lehet elhelyezni, ami azt jelenti, hogy ha az (X, X) ponton átmenő, X-tengellyel párhuzamos egyenest elhelyeztünk, akkor el kell helyezni a párját, tehát a (X, X) ponton átmenő, Y-tengellyel párhuzamos egyenest is.

Írj programot, amely

A. kiszámítja, hogy legkevesebb hány tengelyekkel párhuzamos egyenes-pár kell az összes pont lefedésére,

B. és megad minimális számú lefedő egyenes-párokat!

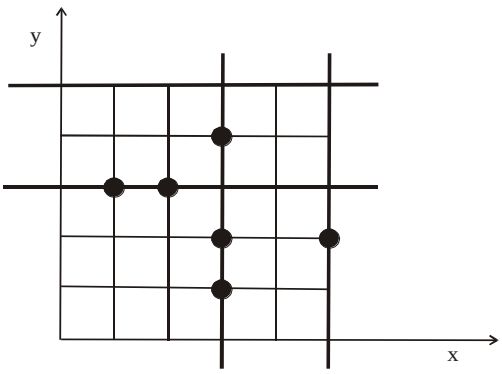
A LEFED.BE állomány első sora az N és M két egész számot tartalmazza: a négyzetrács sorainak és oszlopainak a számát ($1 \le N \le 200$) és a lefedendő pontok számát ($1 \le M \le 40\,000$). A következő M sor mindegyike egy lefedendő pont koordinátáit tartalmazza ($1 \le X_i \le N$, $1 \le Y_i \le N$).

A LEFED.KI állomány első sorába a lehető legkevesebb, tengelyekkel párhuzamos egyenes-pár E számát kell írni, amellyel az összes pont lefedhető! A második sor pontosan E egész számot tartalmazzon, azokat az X értékeket, amelyekre az (X, X) ponton átmenő, X-tengellyel és Y-tengellyel párhuzamos lefedő egyenes-párt fektettünk! Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni!

Példa:

LEFED.BE
```
5 7
1 3
2 3
3 3
3 1
3 2
3 4
5 2
```

LEFED.KI
```
2
3 5
```



Tizenegyedik-tizenharmadik osztályosok

1. feladat: Építkezés (16 pont)

Egy építkezés befejezéséhez N különböző munkát kell elvégezni. Minden munka pontosan egy nap alatt teljesíthető. A terv alapján tudjuk, hogy bizonyos munkákat előbb kell elvégezni, mint másokat. Pontosabban, a terv (A, B) párok halmazát tartalmazza, ahol az (A, B) pár azt jelenti, hogy az A

munkát előbb kell elvégezni, mint a B munkát. Vannak olyan speciális munkák is, amelyekből egy nap csak egy végezhető el, ráadásul csak a megadott sorrendben (de nem feltétlenül egymást követő napokon).

Írj programot, amely

A. kiszámítja, hogy legkevesebb hány nap kell az összes munka elvégzéséhez;

B. megadja a munkáknak egy olyan beosztását, amely szerint minden munka a lehető legkorábban végezhető el a követelmények teljesülése mellett!

Az `EPITKEZ.BE` állomány első sora az N, K és T egész számokat tartalmazza. N az elvégzendő munkák száma (1≤N≤200), az egyes munkákat az 1,…,N számokkal azonosítjuk. K (1≤K≤N) a speciális munkák száma. T a tervben megadott megelőzési párok száma (1≤T≤10 000). A második sor pontosan K egész számot tartalmaz, minden szám egy-egy speciális munka sorszáma. A további T sor mindegyikében két egész szám van: A, B (1≤A≤N, 1≤B≤N), ami azt jelenti, hogy az A munkát előbb kell elvégezni, mint a B-t.

Az `EPITKEZ.KI` állomány első sorába egy egész számot kell írni: annak a lehető legrövidebb időnek a napokban mért M hosszát, amely alatt az összes munkát el lehet végezni! Ezt követően az állományban pontosan M sornak kell lennie! Az állomány (I+1)-edik sora azon munkák sorszámát tartalmazza egy-egy szóközzel elválasztva, amelyeket legkorábban az I-edik napon lehet és kell elvégezni! Ha nem lehet a tervben megadott feltételeket teljesíteni, akkor az első sorba 0-t kell kiírni!

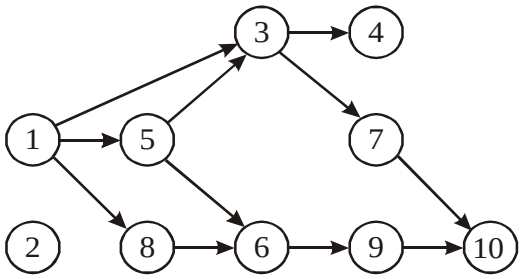
Példa:

`EPITKEZ.BE`

```
10 3 11
1 2 8
1 3
3 4
1 5
5 3
3 7
5 6
1 8
8 6
6 9
9 10
7 10
```

`EPITKEZ.KI`

```
6
1
2 5
8 3
6 4 7
9
10
```



2. feladat: Kép (20 pont)

Adott egy nagy színes raszteres kép (N*N méretű) futamhossz-kódolással, adott továbbá egy kis raszteres kép kódolatlanul (K*K méretű). Futamhossz-kódoláskor a képet sorokra bontjuk, s mindent sort számpárok sorozatával írunk le. A számpár első tagja egy darabszám, a második tagja pedig egy színkód (0 és 255 közötti egész szám), a jelentése pedig: ennyi darab ilyen színű pontot kell egymás mellé tenni. (Például az 1 1 1 1 1 2 2 2 1 1 színkódokat tartalmazó sor futamhossz-kódja: 5 1 3 2 2 1.)

Készíts programot, amely a nagy képben megkeresi a kis kép első előfordulását (fentről lefelé, balról jobbra haladva)!

A `NAGYKEP.BE` állomány első sorában a nagy kép sorainak és oszlopainak a száma (N) van megadva (1≤N≤1000), a következő N sorban pedig a kép egyes sorait leíró futamhossz-kódok (soronként legfeljebb 100 számpár).

A KEP.BE állomány első sorában a kis kép sorainak és oszlopainak a száma (K) van megadva (1≤K≤100), a következő K sor mindegyike K egész számot, az egyes képpontok színkódját tartalmazza.

A KEP.KI állomány egyetlen sorába két egész számot kell írni, SOR-t és OSZLOP-ot, a kis kép első előfordulási pozíciójának koordinátáit a nagy képen belül (1≤SOR≤N-K+1, 1≤OSZLOP≤N-K+1)! Ha a nagy kép nem tartalmazza a kis képet, akkor SOR és OSZLOP legyen 0!

Példa:

NAGYKEP.BE

```
20
20 1
5 1 5 2 5 1 5 2
11 1 1 2 1 3 1 4 1 5 5 2
5 1 5 2 2 1 8 2
5 2 8 1 7 2
11 1 9 2
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
20 1
```

KEP.BE

```
5
1 1 1 1 1
1 2 3 4 5
1 1 2 2 2
1 1 1 2 2
1 2 2 2 2
```

KEP.KI

```
2 11
```

3. feladat: Robot (19 pont)

Egy négyzetrácsos hálózat mezőin elhelyezett tárgyakat egy robotnak kell összegyűjtenie. A robot a bal felső sarokból indul, amelynek (1, 1) a koordinátája, és a jobb alsó sarokba kell érkeznie, amely az (N, N) koordinátájú mező. A robot elemi parancsok sorozatából álló programot tud végrehajtani. Három elemi robotparancs van:

- J : jobbra lép a szomszédos mezőre ((sor,oszlop)-ról (sor,oszlop+1)-re),
- L : lefelé lép a szomszédos mezőre ((sor,oszlop)-ról (sor+1,oszlop)-ra),
- F : felfelé lép a szomszédos mezőre ((sor,oszlop)-ról (sor-1, oszlop)-ra).

A robot nem tud visszafordulni, azaz a programban L után közvetlenül nem állhat F, és F után közvetlenül nem állhat L.

Írj programot, amely feltételezve, hogy a feladat biztosan megoldható

A. kiszámítja, hogy a robot legkevesebb hány lépés megtételével tudja összegyűjteni a tárgyakat;

B. megad egy olyan robotprogramot a tárgyak összegyűjtésére, amely a lehetséges legkevesebb lépést tartalmazza!

A ROBOT.BE állomány első sora a négyzetrácsos hálózat méretét (1≤N≤1000) és a mezőkön található tárgyak számá (1≤K≤100 000) tartalmazza. A további K sor mindegyikében két egész szám van: egy olyan mező koordinátái, ahol tárgy van elhelyezve (1≤SOR≤N, 1≤OSZLOP≤N).

A `ROBOT.KI` állományba két sort kell írni! Az első sor a lehetséges legkevesebb lépés számát (M) tartalmazza, amellyel a robot összegyűjtheti a tárgyakat! A második sor egy olyan robotprogramot írjon le, amelynek végrehajtásával a robot elvégzi a munkát! A sor pontosan M betűből álljon, ahol minden betű egy lehetséges robotparancs jele (J, L vagy F) lehet! A betűk között nem lehet szóköz!

Példa:

`ROBOT.BE`

```
8 10 28
2 2
3 2
5 2
1 5
1 6
3 4
3 6
5 5
6 4
7 6
```

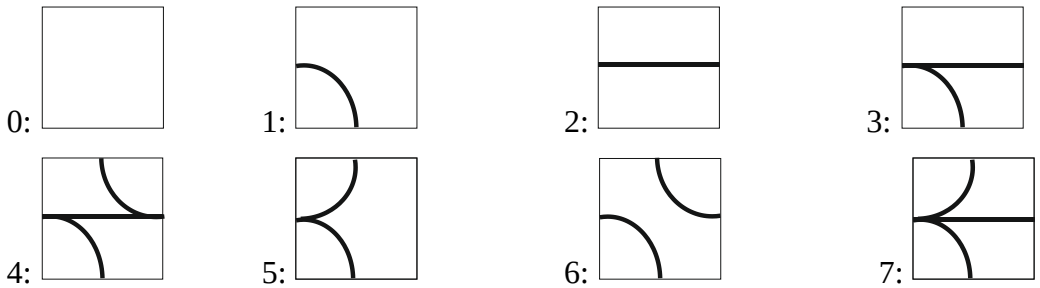
`ROBOT.KI`

```
JLLLLJFFJLLLJFFFFFJLLLLLLJJL
```

	1	2	3	4	5	6	7	8
1					x	x		
2		x						
3		x		x		x		
4								
5		x			x			
6				x				
7						x		
8								

4. feladat: Villamos (20 pont)

Egy négyzetrácsos szerkezetű városban villamosok közlekednek. A villamospályákat egy térképpel írjuk le, ahol minden térképelem egy-egy négyzet alakú pályaelemnek felel meg. A következő pályaelemek vannak:



Az ábrákat úgy kell érteni, hogy a villamos szigorúan csak a kirajzolt pályán mozoghat, azaz ha például az 5-ös pályaelemre felülről lép be, akkor csak úgy mehet ki alul, ha előbb kimegy a szomszédos pályaelemre balra, majd onnan visszajön és lefelé halad tovább. Visszafordulni bármelyik elemről lehet.

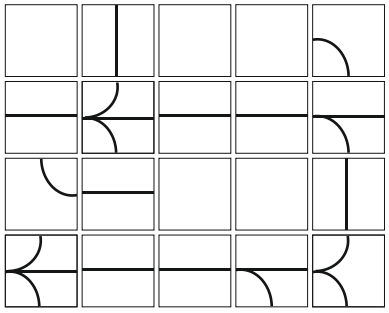
Minden pályaelem négyféle állású lehet:

0: az ábrákon látható 1: 90 fokkal az óramutató járása szerint elforgatva

2: 180 fokkal elforgatva 3: 270 fokkal az óramutató járása szerint elforgatva

Készíts programot, amely megadja, hogy egy adott pályaelemről egy másikra a villamos minimum hány lépésben juthat el!

A `VILLAM.BE` állomány első sorában a térkép sorainak N ($1 \le N \le 100$) és oszlopainak M $1 \le M \le 100$) száma van. A következő N sorban soronként M számjegypár – a pályaelemek kódja – található, egy-egy szóközzel elválasztva. Az egyes sorok a térkép egy-egy sorát írják le. A számjegypárok első tagja a megfelelő ábra száma (0 és 7 közötti érték), a második tagja pedig az elforgatás kódja (0 és 3 közötti érték).



```
00 21 00 00 10
20 70 20 20 30
12 20 00 00 21
70 20 20 30 50
```

Az utolsó sorban 5 egész szám van: a kezdőelem sor- és oszlopindexe, a belépési irány, valamint a célelem sor- és oszlopindexe. A bal felső elem sor- és oszlopindexe (1,1). A kezdőelem biztosan a pálya szélén van, és feltehetjük, hogy a villamos kívülről érkezett. A belépési irány kódja 0, ha balról lépett be a villamos; 1, ha felülről; 2, ha jobbról és 3, ha alulról.

A `VILLAM.KI` állomány egyetlen sorába azt a minimális lépésszámot (az egyik elemről a másikra való átlépések számát) kell írni, amely alatt a villamos eljuthat a kezdőelemről a célelemre! Ha nincs út a két elem között, akkor az állományba –1-et kell írni!

Példa:

VILLAM.BE	VILLAM.KI
4 5	12
00 21 00 00 10	
20 70 20 20 30	
12 20 00 00 21	
70 20 20 30 50	
1 2 1 4 1	

Megjegyzés:

Egy lehetséges út: (1,2),(2,2),(2,1),(2,2),(2,3),(2,4),(2,5),(3,5),(4,5),(4,4),(4,3),(4,2),(4,1)

A verseny végeredménye:

I. korcsoport

1. Acsai Péter	Petőfi Sándor Általános Iskola, Nagykőrös
2. Paulin Roland	Fazekas Mihály Gimnázium, Budapest
Vincze János	Fazekas Mihály Gimnázium, Debrecen
4. Soltész Zoltán	Várkonyi István Általános Iskola, Cegléd
5. Nikházy László	Kazinczy Ferenc Gimnázium, Győr
6. Kormányos Balázs	Radnóti Miklós Gimnázium, Szeged
7. Kunovszki Péter	Kisfaludy Károly Gimnázium, Mohács
8. Tanyi Attila	Eötvös József Gimnázium, Tiszaújváros
9. Izsó Benedek	Veres Péter Gimnázium, Budapest
10. Kiss Attila	Földes Ferenc Gimnázium, Miskolc
Nagy Gergely	Váci u. 43. Általános Iskola, Budapest

II. korcsoport

1. Sztupák Szilárd	Herman Ottó Gimnázium, Miskolc
2. Rácz Béla András	Fazekas Mihály Gimnázium, Budapest
3. Hubai Tamás	Fazekas Mihály Gimnázium, Budapest
4. Ludányi Ákos	Közgazdasági Szakközépiskola és Gimnázium, Eger
5. Kaposi Ambrus	Bencés Gimnázium, Pannonhalma
Tassy Gergely	Veres Péter Gimnázium, Budapest
Sápi Dénes	Földes Ferenc Gimnázium, Miskolc
8. Jancsó Sándor	Árpád Vezér Gimnázium, Sárospatak
Kotyug Gergely	Krúdy Gyula Gimnázium, Nyíregyháza
Medveczky Ádám	Bajza József Gimnázium, Hatvan

III. korcsoport

1. Pelládi Gábor	Földes Ferenc Gimnázium, Miskolc
2. Kerékfy Péter	Fazekas Mihály Gimnázium, Budapest
Károly Dávid	Alternatív Közgazdasági Gimnázium, Budapest
4. Pallos Péter	Fazekas Mihály Gimnázium, Budapest
5. Marton József	Ságvári Endre Gimnázium, Szeged
6. Csordás Hunor	Fazekas Mihály Gimnázium, Budapest
7. Bárkai János	Berzsenyi Dániel Gimnázium, Budapest
8. Bergmann Gábor	Berzsenyi Dániel Gimnázium, Budapest
9. Csóka Endre	Fazekas Mihály Gimnázium, Debrecen
10. Simon Balázs	Révai Miklós Gimnázium, Győr

2003. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendezés (25 pont)

A Comenius Logo egyik játékprogramjában békákat kell sorba rakni úgy, hogy lépésenként kijelölhetjük, hogy melyik béka ugorjon. Ugorni vagy csak szomszédos zsombékra lehet, vagy egy békát lehet átugrani. Kezdetben a jobboldali zsombék üres, s a sorba rakás után bármelyik (akár középen is) lehet üres.

Add meg, hogy minimálisan hány ugrás szükséges ahhoz, hogy az alábbi ábrán látható békákat sorbarendezzük!

A. 

B. 

C. 

D. 

E. 

Példa:

Kezdőállapot: 

1. lépés: 

2. lépés: 

Utolsó lépés: 

2. feladat: Keresés (20 pont)

Az alábbi algoritmusok az X táblázatban (N*M-es) keresnek meg egy Y értéket, a VAN változó igaz értékű lesz, ha megtalálható az X-ben az Y, s hamis, ha nem! Az algoritmusokban a különbség csak a belső ciklus feltételében van (> jel, vagy ≠ jel), illetve a belső ciklus után két algoritmusban szerepel az ***i:=N+1*** értékadás.

Legyen Y tetszőleges 1 és 25 közötti érték! Az alábbi táblázatok esetén melyik algoritmus találja meg biztosan és melyik nem?

A1.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

A3.

1	3	5	7	10
2	14	6	8	23
11	4	17	20	9
12	15	22	21	25
13	16	19	18	24

A5.

1	9	5	7	3
2	4	10	8	6
11	23	17	20	14
12	15	18	21	24
13	16	19	22	25

A2.

4	3	2	1	5
6	7	8	9	10
11	13	12	14	15
16	19	18	17	20
24	22	23	21	25

A4.

1	3	5	7	9
2	4	6	8	10
11	14	17	20	23
12	15	18	21	24
13	16	19	22	25

```
Alfa:
  i:=1; VAN:=hamis
  Ciklus amíg i≤N és nem VAN
    Ha Y≤X(i,M) akkor j:=M
                      Ciklus amíg j≥1 és X(i,j)>Y
                        j:=j-1
                      Ciklus vége
                      VAN:=j≥1 és X(i,j)=Y; i:=N+1
    Elágazás vége
    i:=i+1
  Ciklus vége
Eljárás vége.
```

```
Béta:
  i:=1; VAN:=hamis
  Ciklus amíg i≤N és nem VAN
    Ha Y≤X(i,M) akkor j:=M
                      Ciklus amíg j≥1 és X(i,j)>Y
                        j:=j-1
                      Ciklus vége
                      VAN:=j≥1 és X(i,j)=Y
    Elágazás vége
    i:=i+1
  Ciklus vége
Eljárás vége.
```

```
Gamma:
  i:=1; VAN:=hamis
  Ciklus amíg i≤N és nem VAN
    Ha Y≤X(i,M) akkor j:=M
                      Ciklus amíg j≥1 és X(i,j)≠Y
                        j:=j-1
                      Ciklus vége
                      VAN:=j≥1; i:=N+1
    Elágazás vége
    i:=i+1
  Ciklus vége
Eljárás vége.
```

```
Delta:
  i:=1; VAN:=hamis
  Ciklus amíg i≤N és nem VAN
    Ha Y≤X(i,M) akkor j:=M
                      Ciklus amíg j≥1 és X(i,j)≠Y
                        j:=j-1
                      Ciklus vége
                      VAN:=j≥1
    Elágazás vége
    i:=i+1
  Ciklus vége
Eljárás vége.
```

3. feladat: Szűrés (23 pont)

Az alábbi három algoritmus az egyetlen sort tartalmazó X szöveges változó karakterei közül bizonyosakat a Z szöveges változóba ír. Add meg, hogy melyik eljárás milyen elemeket, illetve milyen elemsorozatokat nem másol át a Z változóba!

```
Első:
  Z:=''
  Ciklus i=1-től hossz(X)-ig
     Ha X[i]≠' ' akkor Z:=Z+X[i]
  Ciklus vége
Eljárás vége.
```

```
Második:
  Z:=''; L:=igaz
     Ciklus i=1-től hossz(X)-ig
       Ha X[i]≠' ' vagy L akkor Z:=Z+X[i]
    L:=(X[i]≠' ')
  Ciklus vége
Eljárás vége.
```

```
Harmadik:
  Z:=''; L:=hamis
  Ciklus i=1-től hossz(X)-ig
     Ha X[i]≠' ' vagy L
                akkor Z:=Z+X[i]
     L:=(X[i]≠' ')
  Ciklus vége
Eljárás vége.
```

4. feladat: Robot (32 pont)

Egy festő robot négyzetrácsos táblát fest, amely kezdetben üres. A kezdő mezőt mindenképpen befesti. Már befestett mezőre nem léphet újból. Haladása során minden mezőt, amelyre rálép, befest. A haladáshoz pedig a következő szabályokat alkalmazhatja:

A. Megvizsgálja annak a mezőnek a szomszédait, amelyiken áll. Ha lehet felfelé lép, ha nem, akkor jobbra, ha ezt sem lehet, akkor lefelé, ha ezt, sem akkor pedig balra.

B. Az aktuális mezőről megpróbál felfelé menni, amíg csak lehet. Ha ez már nem megy, akkor jobbra megy, amíg csak lehet. Ha ez sem sikerül, akkor lefelé megy, amíg csak lehet. Ha ebbe az irányba sem léphetett, akkor balra megy, amíg csak lehet. S ha végül balra sem ment, akkor elölről kezdi a vizsgálatot (azaz újra a felfelé irány következik).

C. Ha tud, akkor egyet felfelé lép. Ha lépett felfelé, ha nem, a következő lépést megpróbálja jobbra tenni. Ha lépett, ha nem, a következőt, akkor is lefelé próbálja. Újra, ha lépett, ha nem, az ezt követőt balra próbálja. Ezután megint a felfelé vizsgálat következik.

D. Megnézi, hogy melyik irányban (felfelé, balra, lefelé, jobbra) van az aktuális mezőtől a legtöbb festetlen mező, s abba az irányba lép, amíg csak haladhat. Ezután újra alkalmazza a szabályt. Ha több irányban is ugyanannyi festetlen mező van, akkor közülük a zárójeles felsorolás szerinti legelsőt kell választani.

E. Megnézi, hogy melyik irányban (felfelé, balra, lefelé, jobbra) van az aktuális mezőtől a legkevesebb festetlen mező, s abban az irányban lép egyet. Ezután újra alkalmazza a szabályt. Ha több irányban is ugyanannyi festetlen mező van, akkor közülük a zárójeles felsorolás szerinti legelsőt kell választani.

Mindegyik eljárás véget ér, ha a robot a lépésszabály alkalmazásával nem tud tovább lépni.

		0		

Töltsd ki a mellékelt festendő táblázatban, hogy a robot melyik mezőre hányadik lépésben ért, ha kezdetben a 0-val jelölt ponton áll és azt már be is festette!

Példa: (az A szabályra)

9	8	1	2
10	7	0	3
11	6	5	4

Kilencedik-tizedik osztályosok

1. feladat: Többségi csoport (20 pont)

Egy osztály tanulói két különálló baráti csoportot alkotnak. Minden tanuló tudja, hogy ki tartozik az ő csoportjába. Az egyik csoportban biztosan többen vannak, mint a másikban, mert a tanulók száma páratlan. Az osztályfőnök szeretne kiválasztani egy tanulót, aki a többségi csoportba tartozik. Ezért kérdéseket tett fel néhány tanulónak, azt tudakolva, hogy X egy csoportba tartozik-e Y-nal? Minden kérdést és a rá adott választ X Y V hármasok formájában feljegyezte, ahol X és Y egy-egy tanuló sorszáma, a V pedig az I betű, ha a válasz szerint X és Y egy csoportban van, egyébként pedig az N betű. A kérdésekre adott válaszokat összegyűjtötte és ezt felhasználva akar kiválasztani egy tanulót a többségi csoportból.

A. Határozd meg, hogy elegendő információval rendelkezik-e az osztályfőnök ahhoz, hogy biztosan meg tudjon nevezni egy tanulót a többségi csoportból!

B. Ha az A. részfeladatra a válasz igenlő, akkor adj is meg egy tanulót, aki biztosan a többségi csoportban van!

Példa:

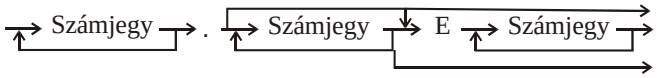
A tanulók száma: 9

```
1 2 N, 4 5 I, 3 6 N, 7 8 I, 5 8 N
A. Igen
B. 9
1. A tanulók száma: 5. 1 2 I, 3 4 I
2. A tanulók száma: 5. 1 2 N, 3 4 N, 4 1 I
3. A tanulók száma: 5. 3 1 I, 1 4 N, 2 5 I
4. A tanulók száma: 7. 1 2 I, 3 4 I, 5 6 I
5. A tanulók száma: 7. 1 2 I, 3 2 I, 4 5 N
6. A tanulók száma: 9. 1 2 I, 3 4 I, 5 6 N, 1 4 I
7. A tanulók száma: 11. 1 2 N, 4 3 N, 1 4 I, 5 6 I, 7 8 I,
                       9 10 N, 9 1 I, 6 7 N
```

8. A tanulók száma: 13. 1 2 I, 4 5 I, 3 10 N, 11 12 I, 10 13 I,
6 7 I, 8 9 N, 6 8 I, 3 4 I, 1 5 I, 11 13 I

2. feladat: REAL nyelv (19 pont)

A REAL nyelven a valós számok speciális formájúak lehetnek, amit az alábbi szintaxisábra ír le:



A. Add meg, hogy az alábbi számok szintaktikusan helyesek-e a fenti definíció szerint:

A1) 11.

A2) .111E1

A3) 11E11

A4) 1.11

A5) 1.E1

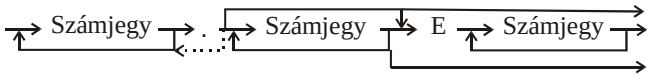
A6) 11

A7) 1.1E11

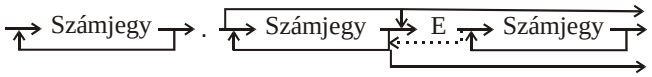
B. Rajzolj le annyiszor újra az ábrát, ahány esetben nem volt helyes a szám, s mindegyikbe rajzold be azt a nyilat, amitől az adott szám felírása helyes lesz!

C. Milyen, a programozási nyelvekben biztosan hibás számokat enged meg az alábbi két szintaxisábra, melyben az eredetihez képest a pontozott vonallal megadott nyilak az újak?

C1.



C2.



3. feladat: Szűrés (23 pont)

Az alábbi három algoritmus az egyetlen sort tartalmazó X szöveges állomány elemeit olvassa, s közülük bizonyosakat a Z szöveges állományba ír. Add meg, hogy melyik eljárás milyen elemeket, illetve milyen elemsorozatokat nem másol át a Z állományba!

```
Első:
  Nyit(X,Z)
  Ciklus amíg nem Vége?(X)
     Olvas(X,c)
     Ha c≠' ' akkor Ír(Z,c)
  Ciklus vége
  Zár(X,Z)
Eljárás vége.
```

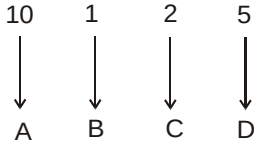
```
Második:
  Nyit(X,Z); L:=igaz
  Ciklus amíg nem Vége?(X)
    Olvas(X,c)
    Ha c≠' ' vagy L akkor Ír(Z,c)
    L:=c≠' '
  Ciklus vége
  Zár(X,Z)
Eljárás vége.
```

```
Harmadik:
  Nyit(X,Z); L:=hamis
  Ciklus amíg nem Vége?(X)
     Olvas(X,c)
     Ha c≠' ' vagy L akkor Ír(Z,c)
     L:=c≠' '
  Ciklus vége
  Zár(X,Z)
Eljárás vége.
```

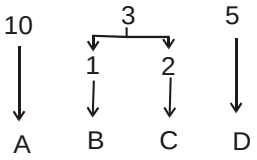
4. feladat: Huffman kód (16 pont)

A karakterek kódja nem minden kódrendszerben áll ugyanannyi bitből. A Huffman kód például úgy készül, hogy leszámoljuk minden betű gyakoriságát a szövegben, a betűket sorba rendezzük gyakoriság alapján, majd ennek alapján a következő algoritmus alapján, mindig a két legkisebb gyakoriságértékűt összevonva, rendelünk hozzá kódokat, és utána újra sorba rendezve gyakoriság szerint.

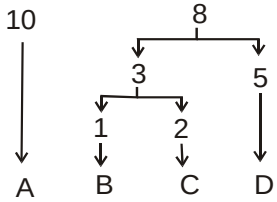
A kezdőhelyzet:



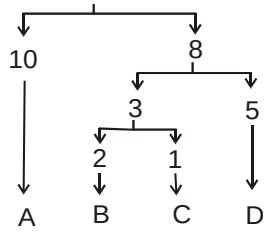
Az 1. lépés után:



A 2. lépés után:



Az utolsó lépés után:



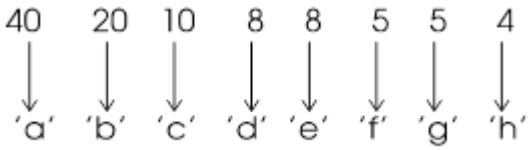
Ezután a karakterkódokat a következőképpen osztjuk ki: az ábrán balra haladva a kód végére 0-t írunk, jobbra haladva pedig 1-et. Így a négy karakter kódja: 'A' – 0, 'B' – 100, 'C' – 101, 'D' – 11.

Legyen a szövegben összesen nyolcféle jel: 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'; a következő gyakoriságokkal:

'a' → 40, 'c' → 10, 'e' → 8, 'g' → 5,
'b' → 20, 'd' → 8, 'f' → 5, 'h' → 4.

Rajzold le lépésenként az ábrákat, majd add meg, hogy melyik betűnek mi lesz a kódja!

A kezdőhelyzet:



5. feladat: FURA nyelv (20 pont)

A FURA nyelvben (a szokásos kettes számrendszer helyett) három jelet (A, B és C betű) használnak a szavak leírására. Mivel ez egy mesterséges nyelv, a szavakat szigorú szabályok betartásával hozzák létre.

- A kiinduló szó mindig az ABB szó.
- Bármely A-val kezdődő szó A mögötti része megduplázható (Ax..y szóból Ax..yx..y alkotható).
- Szó végi B betű helyébe C írható.
- BC szó helyére BBCC írható.
- BC szó törölhető.
- BCB szó helyére BB írható.

A. A FURA nyelv szavai-e az AC, ABBCC, ABCCC, ABBBBBB szavak? Amelyik igen, arra add meg, hogy milyen szabályok alkalmazásával kapható az ABB szóból!

B. Fogalmazd meg, milyen szabályoknak kell teljesülnie egy szóra, hogy a FURA nyelv szava legyen!

Példa:

ABBBBCC szava a FURA nyelvnek, mert ABB ⇒ ABBBB⇒ ABBBC ⇒ ABBBBCC

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rendezés (20 pont)

N+1 rekeszben N rendezendő súlyt helyezünk el, így egyik rekesz üresen marad. A súlyokat úgy rendezhetjük, hogy az üres rekeszbe tesszük valamelyik rekeszbeli súlyt. A rendezés véget ér, ha a súlyok növekvő sorrendben vannak, s közöttük bárhol lehet az egyetlen üres rekesz.

Add meg, hogy az alábbi kiinduló állapotokból minimum hány lépéssel lehet eljutni a rendezett állapothoz!

A:	5	4	3	2	1	üres
B.	2	3	4	5	1	üres
C.	5	3	1	2	4	üres
D.	5	2	3	4	1	üres
E.	1	4	3	2	5	üres

Példa:

Kezdet:	3	2	1	üres
1. lépés:	üres	2	1	3
2. lépés:	1	2	üres	3

2. feladat: Halmazok (20 pont)

Az alábbi algoritmusok az A(1..N) és a B(1..M) növekvően rendezett sorozatok alapján határozzák meg a C(1..K) rendezett sorozatot.

```
Első:
  A(N+1):=max(A(N),B(M))+1; B(M+1):=A(N+1); I:=1; J:=1; K:=0
  Ciklus amíg I<N+1 vagy J<M+1
    K:=K+1
    Ha A(I)<B(J) akkor C(K):=A(I); I:=I+1
    különben ha A(I)>B(J) akkor C(K):=B(J); J:=J+1
    különben C(K):=A(I); I:=I+1; J:=J+1
  Ciklus vége
Eljárás vége.
```

```
Második:
  A(N+1):=max(A(N),B(M))+1; B(M+1):=A(N+1); I:=1; J:=1; K:=0
  Ciklus amíg I<N+1 és J<M+1
    Ha A(I)<B(J) akkor I:=I+1
    különben ha A(I)>B(J) akkor J:=J+1
    különben K:=K+1; C(K):=A(I); I:=I+1; J:=J+1
  Ciklus vége
Eljárás vége.
```

```
Harmadik:
  A(N+1):=max(A(N),B(M))+1; B(M+1):=A(N+1); I:=1; J:=1; K:=0
  Ciklus amíg I<N+1
    Ha A(I)<B(J) akkor K:=K+1; C(K):=A(I); I:=I+1;
    különben ha A(I)>B(J) akkor J:=J+1
    különben I:=I+1; J:=J+1
  Ciklus vége
Eljárás vége.
```

```
Negyedik:
  A(N+1):=max(A(N),B(M))+1; B(M+1):=A(N+1); I:=1; J:=1; K:=0
  Ciklus amíg I<N+1 vagy J<M+1
    Ha A(I)<B(J) akkor K:=K+1; C(K):=A(I); I:=I+1
    különben ha A(I)>B(J) akkor K:=K+1; C(K):=B(J); J:=J+1
    különben I:=I+1; J:=J+1
  Ciklus vége
Eljárás vége.
```

A. Melyik eljárás milyen értékeket tesz a C vektorba, ha feltehetjük, hogy A(i)<A(i+1) és B(i)<B(i+1) minden i-re? Fogalmazd meg általánosan, valamint az A=(1,3,5), B=(2,3,4) példára is!

B. Melyik eljárás milyen értékeket tesz a C vektorba, ha csak azt tehetjük fel, hogy A(i)≤A(i+1) és B(i)≤B(i+1) minden i-re? Fogalmazd meg általánosan, valamint az A=(1,3,3,3,5), B=(2,3,3,4) példára is!

3. feladat: Szűrés (20 pont)

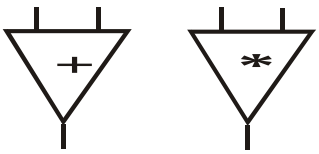
Az alábbi három algoritmus az egyetlen sort tartalmazó X szöveges állomány karaktereit olvassa, s közülük bizonyosakat a Z szöveges állományba ír. Add meg, hogy melyik eljárás milyen karaktereket, illetve milyen karaktersorozatokat nem másol át a Z állományba!

```
Első:
  Nyit(X,Z); L:=0
  Ciklus amíg nem Vége?(X)
     Olvas(X,c)
     Ha c='(' akkor L:=1
     különben ha c=')' akkor L:=0
     különben ha L=0 akkor Ír(Z,c)
  Ciklus vége
  Zár(X,Z)
Eljárás vége.
```

```
Második:
  Nyit(X,Z); L:=0
  Ciklus amíg nem Vége?(X)
    Olvas(X,c)
    Ha c∈{'(','[','{'} akkor L:=1
    különben ha c∈{')',']','}'} akkor L:=0
    különben ha L=0 akkor Ír(Z,c)
  Ciklus vége
  Zár(X,Z)
Eljárás vége.
```

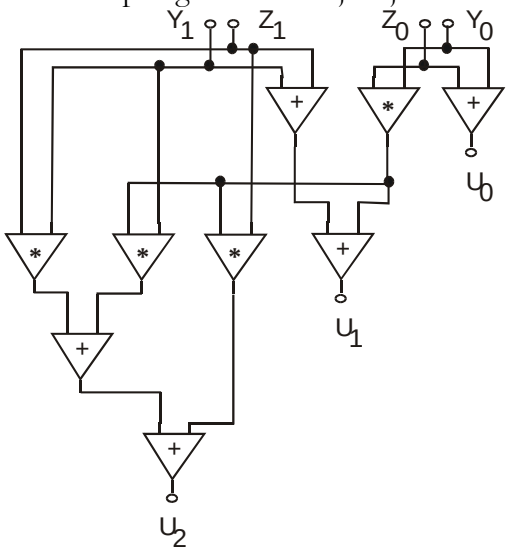
```
Harmadik:
  Nyit(X,Z); L:=0
  Ciklus amíg nem Vége?(X)
    Olvas(X,c)
    Ha c='(' és L=0 akkor L:=1
    különben ha c='[' és L=0 akkor L:=2
    különben ha c='{' és L=0 akkor L:=4
    különben ha c=')' és L=1 akkor L:=0
    különben ha c=']' és L=2 akkor L:=0
    különben ha c='}' és L=4 akkor L:=0
    különben Ha L=0 akkor Ír(Z,c)
  Ciklus vége
  Zár(X,Z)
Eljárás vége.
```

4. feladat: Áramkör (20 pont)



Egy áramkör kétféle alapelemből épül fel. A vezetékek elágazásait festett körrel, bemeneteit és kimeneteit pedig üres körrel jelöljük.

Minden vezetéken 0 vagy 1 értékű jel van. A + jelű alapelem a két bemenete összegét adja modulo 2, a * jelű pedig a két bemenetét összeszorozza.



Az alapelemekből felépítettük a mellékelt hálózatot.

A. Határozd meg a hálózat U_0, U_1, U_2 kimeneteit, ha a bemenetei értéke:

A1. $Y_0=0$, $Y_1=0$, $Z_0=0$, $Z_1=0$,

A2. $Y_0=1$, $Y_1=1$, $Z_0=1$, $Z_1=1$,

A3. $Y_0=1$, $Y_1=0$, $Z_0=0$, $Z_1=1$,

A4. $Y_0=0$, $Y_1=1$, $Z_0=0$, $Z_1=1$.

B. Fogalmazd meg általánosan, hogyan függ az U kimenet az Y,Z bemenettől!

5. feladat: Nyelv (20 pont)

A FURA nyelvben (a szokásos kettes számrendszer helyett) három jelet (A, B és C betű) használnak a szavak leírására. Mivel ez egy mesterséges nyelv, a szavakat az alábbi eljárások tetszőleges sorrendű és számú alkalmazásával állíthatjuk elő a kiinduló ABB szóból (az algoritmusokban SZÓ[x..y] a szó x. és y. betűje közötti részét jelenti, a teljes SZÓ ezek alapján a SZÓ[1..Hossz(SZÓ)] formulával is felírható):

```
Első(SZÓ):
  Ha SZÓ[Hossz(SZÓ)]='B' akkor SZÓ[Hossz(SZÓ)]='C'
Eljárás vége.

Második(SZÓ):
  Ha SZÓ[1]='A' akkor SZÓ:=SZÓ+SZÓ[2..Hossz(SZÓ)]
Eljárás vége.

Harmadik(SZÓ,I):
  Ha 1≤I és I<Hossz(SZÓ) és SZÓ[I]='B' és SZÓ[I+1]='C'
     akkor SZÓ:=SZÓ[1..I]+'BC'+SZÓ[I+1..Hossz(SZÓ)]
Eljárás vége.
```

```
Negyedik(SZÓ,I):
  Ha 1≤I és I<Hossz(SZÓ) és SZÓ[I]='B' és SZÓ[I+1]='C'
     akkor SZÓ:=SZÓ[1..I-1]+SZÓ[I+2..Hossz(SZÓ)]
Eljárás vége.

Ötödik(SZÓ,I):
  Ha 1<I és I<Hossz(SZÓ) és
                  SZÓ[I-1]='B' és SZÓ[I]='C' és SZÓ[I+1]='B'
     akkor SZÓ:=SZÓ[1..I-1]+SZÓ[I+1..Hossz(SZÓ)]
Eljárás vége.
```

A. A FURA nyelv szavai-e az AC, ABBCC, ABCCC, ABBBBBB szavak? Amelyik igen, arra add meg, hogy milyen eljáráshívások állítják elő az ABB szóból!

B. Fogalmazd meg, milyen szabályoknak kell teljesülnie egy szóra, hogy a FURA nyelv szava legyen! (Fordítva, a megadott szabályt teljesítő minden szónak FURA szónak kell lennie!)

Példa:

```
ABBBBCC szava a FURA nyelvnek, mert a
Második(SZÓ); Első(SZÓ); Harmadik(SZÓ, 4) eljáráshívások elő-
állítják ABBBBCC-t, ha a SZÓ kezdeti értéke ABB.
```

2003. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Hangok száma (20 pont)

Egy magyar szóban lehetnek több karakterrel leírt mássalhangzók is (pl. sz, cs, ty, dzs, …). Feltesszük, hogy az egymás melletti s+z, … betűket mindig egy hangnak, azaz sz-nek, … értelmezhetjük. A hosszú mássalhangzókat (pl. ss, ssz,…) egy hangnak kell venni!

Írj programot, amely beolvas egy szót, majd megadja, hogy hány hang van benne!

Példa:

Bemenet:	Kimenet:
kesztyű	5
hosszú	4

2. feladat: Eszperantó számok (27 pont)

Eszperantó nyelven a számokat így írják: 1 – unu, 2 – du, 3 – tri, 4 – kvar, 5 – kvin, 6 – ses, 7 – sep, 8 – ok, 9 – nau, 10 – dek, 100 – cent, 1000 – mil.

A többjegyű számokat a magyarhoz hasonlóan képezik: 11 – dek unu, 12 – dek du, 20 – dudek, 25 – dudek kvin, 40 – kvardek, 167 – cent sesdek sep, 378 – tricent sepdek ok, 2002 – dumil du.

Készíts programot, amely beolvas egy N számot (1≤N≤9999), majd kiírja a képernyőre eszperantó nyelven!

3. feladat: Virág (28 pont)

Egy virágoskert minden parcellájában egy-egy növény található. Ez a növény az első héten kikel (K), a második héten megnő (N), a harmadik héten virágzik (V), a negyedik héten termést érlel (T), az ötödik héten elpusztul (E), de a nyomában a következő héten kikel egy új növény.

Írj programot, amely beolvassa a kert virágai kezdőállapotát, majd megadja, hogy hányadik héten szedhetnénk a legtöbb virágot és mennyit! Ha több héten is ugyanannyi virágot szedhetünk, akkor a legkorábbi hetet adjuk meg.

A program először olvassa be, hogy a kertben a virágok hány sorban (1≤SOR≤20) és hány oszlopban (1≤OSZLOP≤20) helyezkednek el, majd pedig soronként olvassa be az egyes növények állapotát (K,N,V,T,E betűk valamelyike)!

Példa:

```
Bemenet:        Kimenet:
2 3             3 3
EKN
EKK
```

1. hét: 0	2. hét: 1	3. hét: 3	4. hét: 2	5. hét: 0	6. hét: 0
EKN	KNV	NVT	VTE	TEK	EKN
EKK	KNN	NVV	VTT	TEE	EKK

Kilencedik-tizedik osztályosok

1. feladat: Mássalhangzók (12 pont)

Angol szavakban időnként több mássalhangzót is írnak egymás mellé.

Készíts programot, amely megadja az egymás melletti mássalhangzók számát!

A MASSAL.BE állomány egyetlen sorában egy legalább 1 és legfeljebb 255 karakterrel leírt angol szó van.

A MASSAL.KI állományba annyi számot kell írni, amennyi a bemeneti szóban levő mássalhangzó sorozatok száma! Az i-edik szám a szó i-edik csupa mássalhangzóból álló része mássalhangzószáma legyen!

Példa:	MASSAL.BE	MASSAL.KI
1. példa:	computers	1 2 1 2
2. példa:	toast	1 2

2. feladat: Képkódolás (18 pont)

Egy NxN-es színes képet (N kettőhatvány) a következőképpen kódolunk:

- Ha a kép egyszínű, akkor a kódja: 0 szín
- Ha nem egyszínű, akkor bontsuk négy egyforma részre:

1	2
3	4

- Ezzel négy kódrészlet áll elő, a kód első jele a fenti 4 számjegy, s ezután a 4 részre alkalmazzuk újra ugyanezt a módszert.

Példa:

```
5666 kódja: 1105;1206;1306;1406;206;306;4107;4207;4308;4409
6666
6677
6689
```

Írj programot, amely egy adott kódhalmazhoz megadja az általa kódolt képet!

A DEKODOL.BE állomány első sorában a kép mérete (1≤N≤128, N kettőhatvány) és a kódhalmaz elemszáma (1≤M≤N*N) van. A következő M sor mindegyikében egy-egy négyzet alakú tartomány kódja van. A kód nem tartalmaz semmilyen elválasztójelet. A szín jele tetszőleges karakter lehet.

A DEKODOL.KI állományba pontosan N+1 sort kell írni, az első sorba a kép méretét (N), minden további sorában pedig pontosan N jel legyen, a kép egy-egy sora képpontjai színe.

Példa:

```
DEKODOL.BE   DEKODOL.KI   DEKODOL.BE  DEKODOL.KI
4 1          4            4 10        4
0a           aaaa         110a        a666
                          aaaa         1206  6666
                          aaaa         1306  66bb
                          aaaa         1406  6689
                                       206
                                       306
                                       410b
                                       420b
                                       4308
                                       4409
```

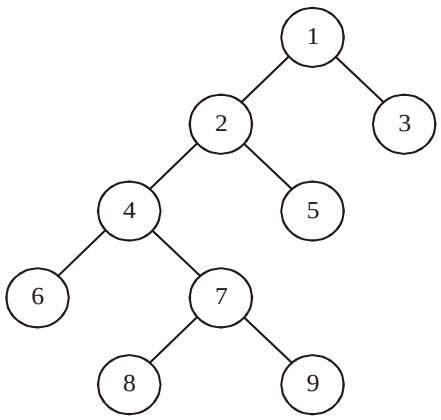
3. feladat: Harmadolás (15 pont)

A Magyarországot elkerülő autópálya építésével megbíztak egy vállalkozót X forintért. A vállalkozó két dolgot tehet: ha el tudja végezni a munkát, akkor a pénzt megtartja magának; ha pedig nem, akkor a munkát és a pénzt három egyenlő részre osztja, egyet megtart, kettőt pedig két új vállalkozónak ad. (Ebből következik, hogy senki sem kaphat kétszer megbízást.) Az újabb vállalkozók ugyanezt a stratégiát követik.

Írj programot, amely megadja, hogy hányan vannak az olyan vállalkozók, akiknél kevesebb pénzt senki sem kapott, azok, akiknél többet senki nem kapott, valamint azok, akik nem adták tovább a munkát másoknak!

A HARMAD.BE állomány első sorában a megbízások (munka- és pénzharmadolások) száma van (1≤N≤1000). A következő N sor mindegyike három számot tartalmaz: a munkát harmadoló vállalkozás sorszámát, valamint a harmadrészt megkapó két újabb vállalkozás sorszámát. Az egyes vállalkozókat sorszámukkal azonosítjuk, az 1-es sorszámú kapja a kiinduló összeget.

A HARMAD.KI állomány első sorába azon vállalkozók számát kell írni, akiknél kevesebb pénzt senki sem kap az autópálya építés során; a második sorba azok számát, akiknél többet nem kap senki, a harmadik sorba pedig azok számát, akik nem adták tovább a munkájukat senkinek! Mind a három sorba a darabszám mögé ki kell írni a megfelelő tulajdonságú vállalkozók sorszámát növekvő sorrendben.



Példa:

```
HARMAD.BE        HARMAD.KI
4                3 7 8 9
1 2 3            2 1 3
2 4 5            5 3 5 6 8 9
4 6 7
7 8 9
```

4. feladat: Konténer rendezés (15 pont)

Egy konténer raktárban N db konténer van egy sorban tárolva. A konténereket el akarják szállítani, ezért mindegyikre rá van írva, hogy melyik városba kell szállítani. A városokat 1-től 4-ig sorszámozzák. A konténereket át kell rendezni úgy, hogy balról jobbra először az 1-essel, majd a 2-essel, aztán a 3-assal, végül a 4-essel jelölt konténerek álljanak. A raktár majdnem tele van, csak az utolsó konténer után van egy konténer számára szabad hely. A rendezést a konténerek fölött mozgatható daruval végezhetjük, amely egy lépésben kiemel a helyéről egy konténert és átteszi azt a szabad helyre, ezzel az átmozgatott konténer helye lesz szabad.

Írj programot, amely kiszámítja, hogy legkevesebb hány lépésben lehet rendezni a konténersort! A rendezés végén a szabad helynek a sor végén kell lennie!

A KONTENER.BE állomány első sorában a konténerek száma van (1≤N≤10 000). A második sor N egész számot tartalmaz. Az i-edik szám annak a városnak a sorszáma (1 és 4 közötti érték), ahova az i-edik konténert szállítani kell.

A KONTENER.KI állomány első és egyetlen sorába a rendezés végrehajtásához minimálisan szükséges lépések számát kell írni!

Példa:

KONTENER.BE	KONTENER.KI
12 1 2 1 3 3 2 2 4 3 4 1 4	7

5. feladat: Verem (15 pont)

A veremautomata olyan gép, amely a bemenetként kapott számsorozaton az alábbi módon működik. Sorban balról jobbra egyesével olvassa a számsorozatot és vagy a sorozat aktuális elemével, vagy a verem tetején lévő elemmel végezhet műveletet. Egy lépésben az alábbi három művelet valamelyikét hajthatja végre:

1. A bemenet aktuális elemét kiírja a kimenetre.
2. A bemenet aktuális elemét beteszi a verembe az ott lévő sorozat elé.
3. A verem tetején lévő (a sorozatban első) elemet kiveszi a veremből és kiírja a kimenetre.

Kezdetben a verem üres. Feladatunkban a veremautomatát arra akarjuk használni, hogy bemenetként kap egy számsorozatot, amely az 1,…N számokat tartalmazza tetszőleges sorrendben, és a kimenetre írja ki az 1,…,M (1≤M≤N) számsorozatot, a lehető legnagyobb M-ig. (A kimenetben minden számnak szerepelnie kell M-ig és sorrendben kell lenniük!)

Írj programot, amely kiszámítja, hogy melyik az a legnagyobb *M* érték, amelyre a veremautomata kimenete az 1,…,M sorozat lehet!

A VEREM.BE szöveges állomány első sorában a bementi sorozat elemszáma van (1≤N≤10 000). A második sor N különböző egész számot tartalmaz. Minden x számra teljesül, hogy 1≤x≤N.

A VEREM.KI állomány első és egyetlen sorába azt a legnagyobb M számot kell írni, amelyre a veremautomata kimenete az 1,…,M sorozat lehet!

Példa:

VEREM.BE	VEREM.KI
10 3 2 1 5 4 6 9 7 10 8	8

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Magánhangzók távolsága (10 pont)

Egy magyar szóban lehetnek több karakterrel leírt mássalhangzók is (pl. sz, cs, dzs, …). Feltesszük, hogy az egymás melletti s+z, … betűket mindig egy hangnak, azaz sz-nek, … értelmezhetjük. A hosszú mássalhangzókat egy hangnak kell venni!

Írj programot, amely megadja, hogy egy szóban a magánhangzók hány hangra vannak egymástól!

A MAGAN.BE állomány egyetlen sorában egy legalább 1 és legfeljebb 255 karakterrel leírt magyar szó van.

A MAGAN.KI állományba eggyel kevesebb számot kell írni, mint a bemeneti szóban levő magánhangzók száma! Az i-edik szám a szó i-edik és azt követő magánhangzója közötti hangok száma legyen!

Példa:

	MAGAN.BE	MAGAN.KI
1. példa:	templomtorony	3 2 1
2. példa:	hosszú	1

2. feladat: Megrendelés (12 pont)

Egy rendezvényt olyan teremben tartanak, ahol M db ülőhely van. Az ülőhelyek 1-től M-ig sorszámozottak. A rendezvény szervezője megrendeléseket fogad. Minden megrendelés egy A B számpárt tartalmaz, ami azt jelenti, hogy a megrendelő olyan ülőhelyet szeretne kapni, amelynek S sorszáma A és B közé esik (A≤S≤B).

Írj programot, amely kiszámítja, hogy a szervező a megrendelések alapján a legjobb esetben hány megrendelést tud kielégíteni és meg is ad egy olyan jegykiosztást, amely kielégíti a megrendeléseket!

A KIOSZT.BE állomány első sorában az ülőhelyek száma (1≤M≤1000)és a a megrendelések száma (1≤N≤1000) van. A következő N sor mindegyike két ülőhely sorszámot tartalmaz (1≤A≤B≤M).

A KIOSZT.KI állomány első sorába a legtöbb kielégíthető megrendelés K számát kell írni! A további K sor tartalmazza a jegykiosztást, minden sor két egész számot tartalmazzon ! Az első szám egy megrendelés sorszáma, a második pedig azon ülőhely sorszáma, amelyet a megrendelő kap! A kiosztás kiírása tetszőleges sorrendben lehet. Ha több megoldás van, akkor egy tetszőlegeset ki lehet írni.

Példa:

KIOSZT.BE	KIOSZT.KI
10 6	4
3 3	5 1
2 2	2 2
2 3	1 3
1 3	6 4
1 2	
2 4	

3. feladat: Lámpák (16 pont)

Egy NxM-es téglalap alakú téren K lámpát helyeztek el. Mindegyiknek ismerjük a helyét. Mindegyik lámpa azt a HxH-s (H páratlan) négyzet alakú területet világítja be, amely átlóinak metszéspontjában áll a lámpa. A világos területek éjszaka is biztonságosak, a sötéteken azonban tanácsosabb nem járni.

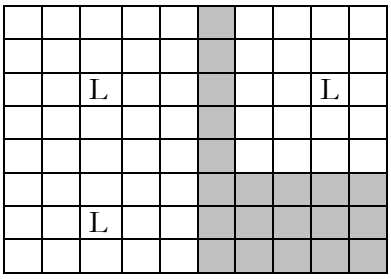
Írj programot, amely megadja, hogy mekkora a téren sötétben maradt terület (a mezők száma), valamint hogy hogyan menjünk át a tér bal felső sarkából a jobb alsó sarkába a legbiztonságosabban úgy, hogy minden pozícióról a 4 oldalszomszédjára léphetünk, átlósan pedig nem léphetünk?

A LAMPAK.BE állomány első sorában a tér sorai (1≤N≤100) és oszlopai száma (1≤M≤100), valamint a lámpák száma (0≤K≤1000) és az általuk bevilágított négyzet oldalhossza (1≤H≤100, H páratlan) van. A következő K sor mindegyike egy lámpa helyét tartalmazza, egy számpárt: közülük az első egy lámpát tartalmazó mező sorindexe, a második pedig az oszlopindexe. A sorokat felülről-lefelé, az oszlopokat balról-jobbra sorszámozzuk.

A LAMPAK.KI állomány első sorába a sötétben maradt mezők számát kell írni. A második sorba azon sötét mezők száma kerüljön, ahányon minimálisan át kell menni, ha a tér bal felső sarkából a jobb alsó sarkába szeretnénk eljutni.

Példa:

```
LAMPAK.BE       LAMPAK.KI
8 10 3 5        20
3 3             4
7 3
3 9
```



4. feladat: Képkódolás (18 pont)

Egy NxN-es színes képet (N kettőhatvány) a következőképpen kódolunk:

Ha a kép egyszínű, akkor a kódja: 0 szín.

1	2
3	4

Ha nem egyszínű, akkor bontsuk négy egyforma részre:

Ezzel négy kódrészlet áll elő, a kód első jele a fenti 4 számjegy, s ezután a 4 részre alkalmazzuk újra ugyanezt a módszert.

Példa:

```
5666        kódja:  1105; 1206; 1306; 1406; 206; 306;
6666                4107; 4207; 4308; 4409
6677
6689
```

Írj programot, amely egy adott képhez kiszámítja a képet megadó kódhalmazt!

A KODOL.BE szöveges állomány első sorában a kép mérete (1≤N≤128, N kettőhatvány) van. A következő N sor mindegyikében pontosan N jel van, egy-egy képsor képpontjainak a színe. A színt tetszőleges karakter jelöli.

A KODOL.KI állomány első sorába a kép méretét (1≤N≤128, N kettőhatvány) és a kódhalmaz elemszámát (1≤M≤1000) kell írni! A következő M sor mindegyikébe egy-egy négyzet alakú tartomány kódját kell írni kód szerint lexikografikusan növekvő sorrendben (lásd a példát)! A kód nem tartalmazhat semmilyen elválasztójelet!

Példa:

```
KODOL.BE    KODOL.KI    KODOL.BE  KODOL.KI
4           4 1         4         4 10
aaaa        0a          abbb      110a
aaaa                    bbbb      120b
aaaa                    bb77      130b
aaaa                    bb89      140b
```

```
20b
30b
4107
4207
4308
4409
```

5. feladat: Szavak (19 pont)

Adott szóra alkalmazott betű-helyettesítésen azt értjük, hogy a szó minden betűjének helyére egy megadott (legalább egy betűből álló) szót írunk úgy, hogy minden betű minden előfordulását ugyanazon szóval helyettesítjük. Különböző betűk különböző szavakkal helyettesíthetőek. Adott szónak adott betű-helyettesítés melletti képén azt a szót értjük amelyet a helyettesítés elvégzésével kapunk.

Írj programot, amely kiszámítja, hogy van-e olyan betű-helyettesítés, amely mellett két adott szó képe megegyezik!

A SZAVAK.BE állomány első két sorában van a két szó, soronként egy-egy. Mindkét szó hossza legfeljebb 33. A szavak csak az angol ábécé nagybetűit (A-tól Z-ig) tartalmazhatják. A két szóban pontosan ugyanazok a betűk fordulnak elő.

A SZAVAK.KI állomány első sorába egy L egész számot kell írni! L értéke 0 legyen, ha nincs olyan betű-helyettesítés, amely mellett a két szó egybeesik! Egyébként L a legrövidebb olyan szó hossza, amire van olyan betű-helyettesítés, hogy a két szó képe megegyezik és hossza L! A további sorokban meg kell adni a betű-helyettesítést, annyi sort kell kiírni, ahány különböző betű szerepelt a két bemeneti szóban! Minden sor első karaktere a helyettesítendő betű legyen, majd egy szóközzel elválasztva álljon az a szó, amelyre a betűt helyettesítjük! A sorokat tetszőleges sorrendben lehet kiírni.

Ha a programod nem a legkisebb ilyen L-et számítja ki, de a helyettesítéssel a két szó egybeesik, akkor fél pontszámot kapsz a megoldásra.

Példa:

SZAVAK.BE	SZAVAK.KI
BAACBD	7
ABDCD	A A
	B A
	C A
	D AA

2003. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Naptár (24 pont)

A keresztény országokban nagyon sokáig az ún. *Julianus-naptár*t használták, amelyben minden négygyel osztható év 366 napos szökőév volt. A valódi, ún. *tropikus* év ennél egy kicsivel rövidebb, így XIII. Gergely pápa utasítására kidolgozták az ún. *Gergely-naptár*t. Ebben a 100-zal osztható évek közül csak a 400-zal is oszthatók szökőévek. Az új naptár bevezetésekor úgy döntöttek, hogy az 1582. október 4. után azonnal október 15. következzen. A korábbi dátumokat így nem változtatták meg, de ettől kezdve az új naptár szerint számoltak.

Készíts programot, amely beolvas egy 1 és 2003 közötti dátumot a Julianus-naptár szerint, majd kiírja, hogy ez a nap milyen dátumú a Gergely-naptár szerint!

Példa:

```
Bemenet:                Kimenet:
1582 10 4               1582 10 4
1582 10 5               1582 10 15
1700 2 18               1700 2 28
1700 2 19               1700 3 1
```

2. feladat: Verseny (25 pont)

Egy futóversenyen a versenyzőket (N db) rajtszám szerint (a rajtszám 1-től N-ig fut) percenként indítják. A célba érkezési listán időrendben megadják, hogy melyik rajtszámú versenyző mikor érkezett célba.

Írj programot, amely beolvassa a versenyzők számát (1≤N≤100), a célba érkezett versenyzők számát (1≤M≤N), majd M darab versenyző sorszámot (1≤sorszám≤N) és célba érkezési időt (0 és 10 000 közötti egész számok, növekvő sorrendben). A program ezek alapján készítse el az eredménylistát, valamint adja meg, hogy kik nem érkeztek célba.

Példa:

```
Bemenet:                Kimenet:
N=6, M=4                1. helyezett: 5
sorszám: 2 idő: 10      2. helyezett: 2 4
sorszám: 4 idő: 12      4. helyezett: 1
sorszám: 5 idő: 12
sorszám: 1 idő: 15      Nem érkezett célba: 3 6
```

3. feladat: Kártya (26 pont)

Francia kártyával sokféle játékot játszhatunk. Az egyik játékban a játékosok az osztásnál kapott 14 lapból letehetnek kártyahármasokat: vagy egymást követő azonos színű lapokat (pl. pikk 9 10 bubi), vagy pedig azonos értékű lapokat (pl. pikk 3 kör 3 treff 3).

Készíts programot, amely egyesével beolvassa egy játékos treff, pikk, kőr, valamint káró színű kártyáit növekvő sorrendben, majd megadja, hogy milyen kártyahármasokat tehet le!

A kártyák színe: *treff*, *pikk*, *kőr*, *káró*; értéke pedig *2*, *3*, *4*, *5*, *6*, *7*, *8*, *9*, *10*, *bubi*, *dáma*, *király*, *ász* lehet.

Példa:

```
Bemenet:                Kimenet:
treff: 2 6 7 8          treff 6 7 8
pikk: 8 9 10            pikk 8 9 10
kőr: 9 10 dáma          pikk 10 kör 10 káró 10
káró: 2 5 10 ász
```

Kilencedik-tizedik osztályosok

1. feladat: Barátok (15 pont)

Egy osztályba N tanuló jár. Ismerünk tanulópárokat, akik barátai egymásnak. A barátság ún. tranzitív kapcsolat, azaz ha A barátja B-nek és B barátja C-nek, abból következik hogy A is barátja C-nek. A barátság kapcsolat szimmetrikus is, azaz, ha A barátja B-nek, akkor B is barátja A-nak.

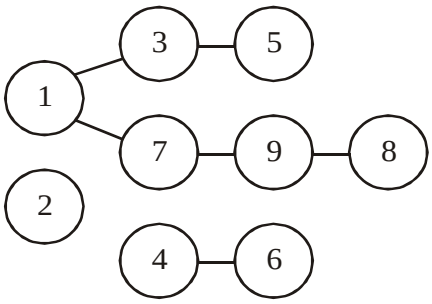
Írj programot, amely megadja, hogy az osztály hány baráti csoportra bontható!

A BARATOK.BE állomány első sorában a tanulók száma (2≤N≤100) és az ismert baráti kapcsolatok száma (0≤M≤10 000) van. A következő M sor mindegyikében egy-egy számpár van, két tanuló sorszáma, akikről tudjuk, hogy barátok.

A BARATOK.KI állomány első sorába azt a K számot kell írni, ahány baráti csoportra az osztályt bontani lehet! Ha tudjuk, hogy A és B barátok, akkor ugyanazon csoportba kell tartozniuk, ha pedig nem barátok, akkor különbözőbe! Mindegyikbe a baráti csoportba tartozó tanulók sorszámát kell írni, növekvő sorrendben! A sorokat a csoport legkisebb sorszámú tagja szerint növekvően kell kiírni!

Példa:

BARATOK.BE	BARATOK.KI
9 6	3
1 3	1 3 5 7 8 9
3 5	2
4 6	4 6
7 9	
8 9	
1 7	



2. feladat: Szállítás (15 pont)

Egy vállalat termeléssel, raktározással és árusítással is foglalkozik. Összesen N telephelye van, s egyiken sem foglalkozik kétféle tevékenységgel. Ismerjük, hogy mely telephelyről mely telephelyre szállít árut. Szállítási útvonalai lehetnek: termelő helyről raktározó vagy árusító helyre; raktározó helyről másik raktározó vagy pedig árusító helyre.

Készíts programot, amely megadja

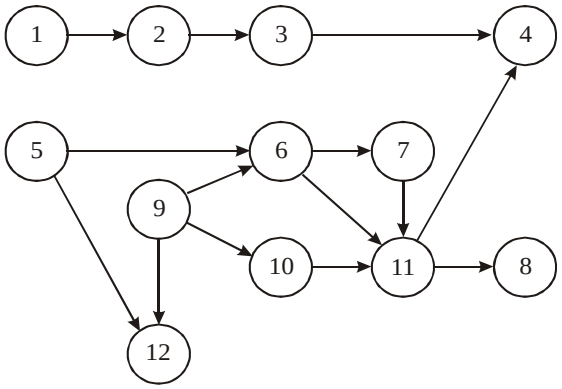
A. azokat a telephelyeket, amelyek termeléssel foglalkoznak;
B. azokat a telephelyeket, amelyek árusítással foglalkoznak;
C. azokat az árusító telephelyeket, ahova csak termelő telephelyről küldenek árut;
D. azokat a raktározó telephelyeket, akiknek nincs kapcsolatuk közvetlenül sem termelővel, sem árusítóval;
E. azokat a raktározó telephelyeket, amelyeknek sem összegyűjtő, sem szétosztó funkciójuk nincs!

A SZALLIT.BE állomány első sorában a telephelyek száma (1≤N≤100) és a szállítási útvonalak száma (1≤M≤1000) van. A következő M sor mindegyikében két telephely sorszáma (1≤A≠B≤N) van, ami azt jelenti, hogy az A telephelyről visznek árut a B telephelyre.

A SZALLIT.KI állományba 5 sort kell írni! Az első sorba az A, a másodikba a B, a harmadikba a C, a negyedikbe a D, az ötödikbe pedig az E részfeladat megoldása kerüljön! (Ha valamelyik részfeladat megoldása nincs kész, a hozzá tartozó üres sort akkor is ki kell írni az eredménybe!) Mindegyik sorba a megfelelő tulajdonságú telephelyek számát kell írni, majd pedig a megfelelő telephelyek sorszámát növekvő sorrendben!

Példa:

```
SZALLIT.BE          SZALLIT.KI
12 14               3 1 5 9
1 2                 3 4 8 12
2 3                 1 12
3 4                 1 7
5 6                 4 2 3 7 10
5 12
6 7
6 11
7 11
9 6
9 10
9 12
10 11
11 4
11 8
```



3. feladat: Titkos társaság (15 pont)

Egy titkos társaság hierarchikusan épül fel, minden tagja csak a felettesét és a hozzá közvetlenül beosztottakat ismeri. A társaságnak pontosan egy olyan tagja van, akinek nincs főnöke. Bármelyik tag küldhet levelet bármelyik tagnak. A szervezeten belül a levelek továbbítása úgy történik, hogy egy lépésben vagy a közvetlen főnökhöz, vagy a közvetlen beosztotthoz továbbítódik a levél.

Írj programot, amely adott két, X és Y tagra kiszámítja az alábbi kérdésekre adandó választ!

A. ha csak a beosztottaknak lehet levelet küldeni, akkor az egyik küldhet-e levelet a másiknak;

B. ha beosztottaknak és feletteseknek is lehet levelet küldeni, akkor az X-től Y-nak küldött levél hány lépés alatt ér el Y-hoz;

C. hány beosztottja –nem csak közvetlen– van az X és az Y tagnak?

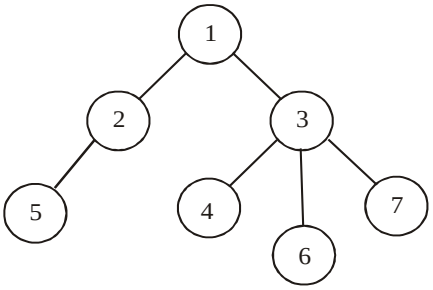
A TITKOS.BE állomány első sorában a társaság tagjainak száma ($1 \le N \le 1000$), valamint a két ember sorszáma ($1 \le X,Y \le N$) van. A következő N-1 sorban egy-egy felettesi kapcsolatot leíró (P,Q) számpár ($1 \le P,Q \le N$, $P \ne Q$) van, jelentése: P felettese Q-nak. Mindenkinek pontosan egy felettese van, kivéve a társaság nagyfőnökét (1-es sorszámú), akinek nincs felettese.

A TITKOS.KI állomány első sorába az A, második sorába a B, harmadik sorába a C kérdésre adott választ kell írni!

Az első sorba egyetlen számot kell írni: 1-et, ha A küldhet levelet B-nek; 2-t, ha B küldhet levelet A-nak; egyébként pedig 3-at! A második sorba egyetlen számot kell írni: hány lépés alatt juthat el a levél egyiktől a másikig! A harmadik sorba két számot kell írni, X és Y beosztottjainak a számát!

Példa:

```
TITKOS.BE           TITKOS.KI
7 2 3               3
1 2                 2
2 5                 1 3
1 3
3 4
3 6
3 7
```



4. feladat: Rendőrök (15 pont)

Egy autópálya mentén N város helyezkedik el. Bizonyos városokban autópálya rendőrök tartózkodnak, némelyikben több is, némelyikben egy sem. Összesen legalább N rendőr van. Azt szeretnénk elérni, hogy minden városban legyen legalább egy rendőr, ezért át kell csoportosítani. Az átcsoportosítást a lehető legkisebb összköltséggel kell végrehajtani. Egy rendőr i. városból a j-edikbe történő átmozgatásának költsége a várossorszámok különbségének abszolút értéke: |i-j|.

Írj programot, amely kiszámítja az átcsoportosítás lehető legkisebb összköltségét!

A RENDOR.BE állomány első sorában a városok száma (1≤N≤1000) van. A második sorban pontosan N szám van: az i-edik szám azt adja meg, hogy az i-edik városban kezdetben hány rendőr tartózkodik. Összesen legalább N rendőr van a városokban.

A RENDOR.KI állomány első és egyetlen sorába azt a legkisebb összköltséget kell írni, amellyel elérhető, hogy minden városban legyen rendőr!

Példa:

RENDOR.BE	RENDOR.KI
7	5
0 1 0 3 2 0 1	

5. feladat: Futó (15 pont)

Egy útvonalon a céltól különböző távolságra egyszerre indul N futó. A futók egyenletes sebességgel futnak, az i-edik futó másodpercenként Ai centimétert tesz meg. Azt mondjuk, hogy a távolabbról induló lehagyja a közelebbről indulót, ha van olyan időpont, amikor a távolabbról jövő közelebb kerül a célhoz, mint a közelebbről jövő.

Írj programot, amely megadja azokat az időpontokat, amikor a sorrend változott, azaz egy távolabbról induló lehagyott egy közelebbről indulót!

A FUTO.BE állomány első sorában a futók száma (1≤N≤100) van. A következő N sor mindegyike két egész számot tartalmaz: T_i az i-edik futó rajthelyének távolsága a céltól, A_i pedig az a távolság, amit 1 másodperc alatt megtesz. ($T_1 \le 1\,000\,000$, $T_i > T_{i+1}$, $T_n > 0$, $A_i > 0$).

A FUTO.KI állomány egyetlen sorába azon időpontok K számát kell írni, amikor a sorrend változik, utána pedig a K megfelelő időpontot!

Példa:

```
FUTO.BE                FUTO.KI
5                      4 4 5 7 13
100 5
96 6        Magyarázat: (versenyzők távolsága a céltól)
88 4                0.  1.  2.  3.  4.  5.  6.  7. … időpont
81 2        1:  100  95  90  85  80  75  70  65 …
10 1        2:   96  90  84  78  72  66  60  54 …
            3:   88  84  80  76  72  68  64  60 …
            4:   81  79  77  75  73  71  69  67 …
            5:   10   9   8   7   6   5   4   3 …
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Régió (15 pont)

Egy megyén belül a településeket régiókba szeretnék csoportosítani. Ismerjük az egyes települések koordinátáit. Két település távolságán a koordináta-különbségeik abszolút értékének összegét értjük, azaz TÁVOLSÁG((x,y),(a,b))=|x-a|+|y-b|.

Két települést azonos régióba teszünk, ha egyikből a másikba el lehet jutni a régió településein keresztül úgy, hogy az egymást követő települések távolsága legfeljebb T kilométer.

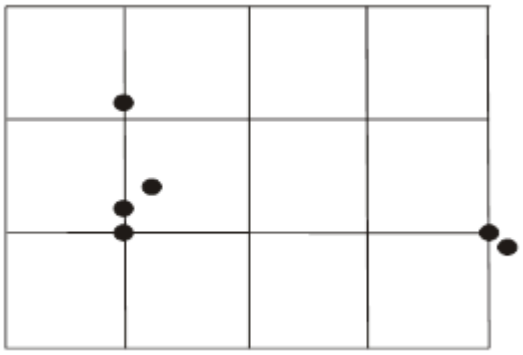
Írj programot, amely megadja, hogy a települések hány régiót alkotnak, és mely települések tartoznak egy régióba!

A REGIO.BE szöveges állomány első sorában a városok száma (2≤N≤100) és a régióba kerülés határát jelentő távolság (1≤T≤100) van. A következő N sor mindegyikében egy-egy számpár van, az adott város x- és y-koordinátája (0 és 1000 közötti egész számok).

A REGIO.KI szöveges állomány első sorába a legkisebb K számot kell írni, ahány régióba lehet besorolni a településeket! A következő K sorba az egyes régiókat kell írni, tetszőleges sorrendben! Egy sorba a régióba tartozó települések sorszámát kell írni, egy-egy szóközzel elválasztva, növekvő sorrendben!

Példa:

```
REGIO.BE          REGIO.KI
6 50              3
100 100           1 3 5
100 220           2
100 120           4 6
300 100
120 140
310 90
```



2. feladat: Repülőút (16 pont)

Egy repülőtársaság N város között üzemeltet járatokat. A városokat a természetes számokkal azonosítják 1-től N-ig. A társaság jelentős kedvezményt ad, ha az utas olyan útvonalat választ, hogy az utazás során mindig nagyobb sorszámú városba megy. Az 1. városból szeretnénk eljutni az N. városba kedvezményes útvonalon.

Írj programot, amely megadja azokat a városokat, amelyeken mindenképpen át kell haladnunk, valamint azokat a várospárokat, amelyek közötti járatot mindenképpen igénybe kell venni bármely kedvezményes útvonalon akarunk az 1. városból az N. városba jutni!

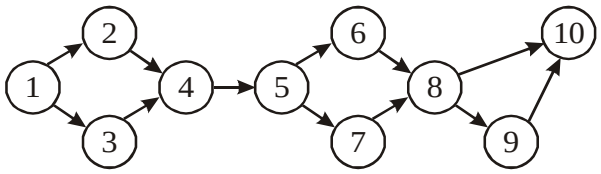
A REPUT.BE állomány első sorában a városok száma (1≤N≤100) és a járatok száma (1≤M≤1000) van. A következő M sor mindegyikében egy-egy számpár (1≤P<Q≤N) van. Ez azt jelenti, hogy van járat a P és a Q város között. Az 1. városból bármely másik városba el lehet jutni és bármely városból el lehet jutni az N. városba alkalmas járatokkal.

A REPUT.KI állomány első sorába a kikerülhetetlen városok K számát kell írni, majd ettől egy-egy szóközzel elválasztva a kikerülhetetlen városok sorszámát növekvő sorrendben! A második sorba a kikerülhetetlen járatok M számát kell írni! A következő M sor mindegyikébe egy-egy elkerülhetetlen járatot kell írni, a két város sorszámát!

Példa:

```
REPUT.BE
10 12
1 2
1 3
2 4
3 4
4 5
5 6
5 7
6 8
7 8
8 9
9 10
8 10
```

```
REPUT.KI
3 4 5 8
1
4 5
```



3. feladat: Titkos társaság (15 pont)

Egy titkos társaság hierarchikusan épül fel, minden tagja csak a felettesét és a hozzá közvetlenül beosztott legfeljebb két tagot ismeri. A társaságnak pontosan egy olyan tagja van, akinek nincs főnöke. Bármelyik tag küldhet levelet bármelyik tagnak. Azonban minden levél csak úgy juthat el a feladótól a címzetthez, hogy egy lépésben vagy a közvetlen főnökhöz, vagy közvetlen beosztotthoz továbbítódik.

Írj programot, amely adott két, X és Y tagra kiszámítja az alábbi kérdésekre adandó választ!

A. Hány beosztottja – nem csak közvetlen – van az X és az Y tagnak?

B. Hány lépéssel továbbítódik egy levél, ha X küld levelet Y-nak?

C. Mennyi a legkevesebb lépésszám, ami alatt biztosan odaér egy levél bárki legyen is a feladó, illetve a címzett?

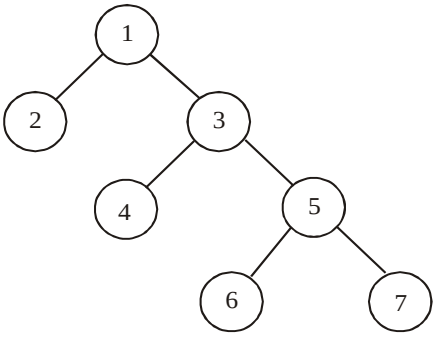
A TARSASAG.BE szöveges állomány első sorában a társaság tagjainak száma (1≤N≤1000), valamint a két tag sorszáma (1≤X,Y≤N) van. A társaság tagjai olyan sorszámot kaptak 1 és N között, hogy mindenkinek nagyobb a sorszáma, mint a közvetlen főnökéé. A második sor pontosan N db 0 és N közötti egész számot tartalmaz. Az i-edik szám a társaság i-sorszámú tagjának közvetlen főnökét adja. A sorban az első szám 0, mivel pontosan egy tagnak, az 1 sorszámúnak nincs főnöke. Minden tag legfeljebb két másik tagnak lehet a közvetlen főnöke.

A TARSASAG.KI szöveges állomány első sorába az A, második sorába a B, harmadik sorába a C kérdésre adott választ kell írni!

Az első sorba az X és az Y tag beosztottjainak a számát kell írni! A második sorba azt a lépésszámot kell írni, amely alatt egy levél eljut az X tagtól a Y taghoz! A harmadik sorba a legkevesebb lépésszámot kell írni, ami alatt biztosan odaér egy levél, bárki legyen is a feladó, illetve a címzett!

Példa:

TARSASAG:BE	TARSASAG.KI
7 2 5	0 2
0 1 1 3 3 5 5	3
	4



4. feladat: Rendőrök (14 pont)

Egy autópálya mentén N város helyezkedik el. Bizonyos városokban autópálya rendőrök tartózkodnak, némelyikben több is, némelyikben egy sem. Összesen legfeljebb N rendőr van. Azt szeretnénk elérni, hogy a lehető legtöbb városban legyen rendőr, ezért át kell csoportosítani. Az átcsoportosítást a lehető legkisebb összköltséggel kell végrehajtani. Egy rendőr i. városból a j.-be történő átmozgatásának költsége a városszámok különbségének abszolút értéke: |i-j|.

Írj programot, amely kiszámítja az átcsoportosítás lehető legkisebb összköltségét és megadja azt, hogy az átcsoportosítás után mely városokban lesz rendőr!

A RENDOR.BE állomány első sorában a városok száma (1≤N≤100) van. A második sorban pontosan N szám van. Az i-edik szám azt adja meg, hogy az i-edik városban kezdetben hány rendőr tartózkodik. Összesen legfeljebb N rendőr van a városokban.

A RENDOR.KI állomány első sorába azt a legkisebb összköltséget kell írni, amellyel elérhető, hogy a legtöbb városban legyen rendőr! A második sorba pontosan N számot kell írni, az i-edik szám 1-es legyen, ha az i-edik városban lesz rendőr az átmozgatás után, egyébként 0! (Ha több megoldás is van, akkor egy tetszőlegeset ki lehet írni!)

Példa:

RENDOR.BE	RENDOR.KI
7	5
0 1 0 3 2 0 0	1 1 1 1 1 1 0

5. feladat: Hálózat (15 pont)

Minden számítógépes hálózat úgy épül fel, hogy csomópontokat kétirányú adatátvitelt biztosító vonal köt össze közvetlenül. Az általunk vizsgált hálózat olyan, hogy minden csomópont legfeljebb három másik csomóponttal van közvetlenül összekötve. A hálózatot úgy alakították ki, hogy megbízható legyen abban az értelemben, hogy bármely két P és Q csomópontja között legyen két olyan útvonal, amelyeknek nincs közös pontja, csak P és Q.

Írj programot, amely két adott csomópontra meghatároz két olyan útvonalat, amelyeknek a végpontok kivételével, nincs közös pontjuk!

A HALOZAT.BE állomány első sorában a csomópontok száma (3≤N≤1000), a csomópontok közötti közvetlen kapcsolatok száma (3≤M≤3000), s a két kérdéses csomópont sorszáma (1≤P≠Q≤N) van. A következő M sor mindegyikében egy-egy számpár (1≤X≠Y≤N) van. Ez azt jelenti, hogy az X és az Y csomópontot kétirányú átvitelt biztosító közvetlen kapcsolat köti össze.

A HALOZAT.KI állomány két sort tartalmazzon, a megadott P és Q csomópontot összekötő két különböző, közös belső pont nélküli útvonalat! Mind P, mind Q szerepeljen az útvonalban! (Ha több megoldás is van, akkor egy tetszőlegeset ki lehet írni.)

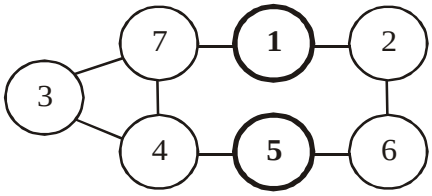
Példa:

HALOZAT.BE

```
7 8 1 5
7 1
4 7
1 2
4 5
3 4
5 6
6 2
3 7
```

HALOZAT.KI

```
1 7 4 5
1 2 6 5
```



A verseny végeredménye:

I. korcsoport

1. Vincze János	Fazekas Mihály Gimnázium, Debrecen
Lóska Ádám	Fazekas Mihály Gimnázium, Debrecen
3. Nagy Gergely	Fazekas Mihály Gimnázium, Budapest
4. Kónya Gábor	Fazekas Mihály Gimnázium, Budapest
5. Beck Róbert	Ságvári Endre Általános Iskola, Oroszlány
6. Gál Péter	Belvárosi Általános Iskola és Gimnázium, Békéscsaba
7. Bálint Bence	SZTE Ságvári Endre Általános Iskola, Szeged
8. Gombos Gergely	Apáczai Csere János Gimnázium, Budapest
9. Szoldatics András	Felsőbüki Nagy Pál Gimnázium, Kapuvár
Orbán Ákos	Petőfi Sándor Gimnázium, Aszód

II. korcsoport

1. Stippinger Marcell	Széchenyi István Gimnázium, Sopron
Láda Ákos	Radnóti Miklós Gimnázium, Dunakeszi
3. Ludányi Ákos	Neumann János Szakközépiskola, Eger
Kovács Máté	Fazekas Mihály Gimnázium, Debrecen
5. Siska Attila	Neumann János Szakközépiskola, Budapest
Lászka Áron	Apáczai Csere János Gimnázium, Budapest
7. Soltész Zoltán	Apáczai Csere János Gimnázium, Budapest
Tassy Gergely	Veres Péter Gimnázium, Budapest
Nikházy László	Kazinczy Ferenc Gimnázium, Győr
10. Acsai Péter	Arany János Református Gimnázium, Nagykőrös
Kormányos Balázs	Radnóti Miklós Gimnázium, Szeged
Hegedűs Tibor	Földes Ferenc Gimnázium, Miskolc

III. korcsoport

1. Móra Péter	Deák Ferenc Gimnázium, Jászárokszállás
2. Pelládi Gábor	Földes Ferenc Gimnázium, Miskolc
3. Bárkai János	Berzsenyi Dániel Gimnázium, Budapest
4. Rendes Gábor	Révai Miklós Gimnázium, Győr
5. Simon Balázs	Révai Miklós Gimnázium, Győr
6. Bergmann Gábor	Berzsenyi Dániel Gimnázium, Budapest
7. Hubai Tamás	Fazekas Mihály Gimnázium, Budapest
Csóka Endre	Fazekas Mihály Gimnázium, Debrecen
Szabó Dávid	Kölcsey Ferenc Gimnázium, Zalaegerszeg
10. Kuderna Csaba	Neumann János Szakközépiskola, Eger

2004. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendezés (30 pont)

Egy osztály tanulói teniszversenyen vesznek részt. N játszma lejátszása után olyan erősorrendet szeretnének felállítani, amelyben senki sem lehet olyan tanuló előtt, aki legyőzte őt. Az (X,Y) pár jelentése: X legyőzte Y-t.

Az alábbi eredmények alapján adj meg egy ilyen sorrendet, illetve ha valahol körbeverés lenne, akkor add meg az abban szereplőket:

A. (A,D), (A,G), (D,B), (G,E), (F,G), (A,B), (B,F), (F,H), (C,H)

B. (A,D), (G,A), (D,B), (G,E), (F,G), (A,B), (B,F), (F,H), (C,H)

C. (A,D), (A,G), (D,B), (G,E), (G,F), (A,C), (B,F), (F,H), (C,H)

D. (A,D), (A,G), (D,B), (G,E), (F,H), (A,B), (B,C), (H,G), (C,F)

E. (A,D), (H,A), (A,G), (D,B), (E,G), (F,G), (A,B), (B,F), (F,H)

F. (A,B), (A,C), (D,B), (G,E), (F,G), (A,D), (B,C), (F,H), (C,H)

Példa:

(A,B), (C,D), (A,C) játszmák esetén az alábbi három mindegyike helyes válasz:

1. A,B,C,D 2. A,C,D,B 3. A,C,B,D

2. feladat: Gépi nyelv (21 pont)

Egy számítógép az alábbi típusú utasításokat érti, és csak az A és B regisztert használhatja:

Utasítás	Jelentése
`ADD A,B`	A:=A+B
`MOV A,B`	B:=A
`INP A`	Beolvasás A-ba
`OUT B`	Kiírás B-ből

Egy szám kétszeresét kiszámító program ezen a nyelven:

```
INP A
ADD A,A
OUT A
```

Készíts programot ezen a nyelven, ami minimális számú utasítással kiszámolja

A. egy szám 16-szorosát;

B. egy szám 10-szeresét;

C. egy szám 13-szorosát!

3. feladat: Túra (24 pont)

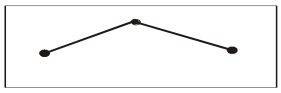
Az osztály kerékpártúrát szervez Budapestről Esztergomba és vissza. Méterenként ismerjük az út tengerszint feletti magasságát (N>2 darab érték a T vektorban). Mivel az emelkedőkön nehéz felfelé haladni, úgy döntöttek, hogy minden emelkedő elején megállnak enni és inni. Az alábbi algoritmus számolja, hány helyen állnak meg a túra során, oda és vissza együttesen.

```
Megállások:
  Ha T(1)<T(2) akkor S:=1 különben S:=0
  Ciklus i=2-től N-1-ig
    Ha T(i-1)=T(i) és T(i)<T(i+1) akkor S:=S+1   (*)
    Ha T(i-1)>T(i) és T(i)<T(i+1) akkor S:=S+2   (**)
    Ha T(i-1)>T(i) és T(i)=T(i+1) akkor S:=S+1   (***)
  Ciklus vége
Eljárás vége.
```

A. Rajzold le, hogy milyen jellegű szakaszok esetén számolunk az egyes elágazás-ágakban (*), (**), (***)!

Például:

```
Ha T(i-1)<T(i) és T(i)>T(i+1) akkor ...
```



B. A (**) sorban miért növeljük kettővel S értékét?

C. Az eljárás időnként jól számolja a pihenők számát, időnként eggyel kevesebbet számol. Mi a feltétele annak, hogy pontosan számoljon és mi annak, hogy eggyel kevesebbet számoljon?

D. Javítsd ki az algoritmust úgy, hogy mindig helyesen számoljon! Ehhez egyetlen sort szabad beszúrnod.

4. feladat: Szótagoló (25 pont)

Az alábbi eljárások egyszerű magyar szavakat szótagolnak. Az A szöveges változóban van a szó, a B szöveges változóba teszik a szótagolt szavat. Egyik sem működik helyesen minden magyar szóra. Melyik milyen típusú szavakra működnek helyesen? Adj egy-egy példát és fogalmazd meg általánosan is!

```
Első(A,B):
  B:=A(1)
  Ciklus i=2-től hossz(A)-ig
    ha A(i) magánhangzó és A(i-1) magánhangzó
         akkor B:=B+'-'+A(i)
      különben B:=B+A(i)
  Ciklus vége
Eljárás vége
```

```
Második(A,B):
  B:=''; C:=''; D:=''
  Ciklus i=1-től hossz(A)-ig
    ha A(i) magánhangzó akkor B:=B+D+C+A(i); C:=''; D:='-'
                      különben C:=C+A(i)
  Ciklus vége
  B:=B+C
Eljárás vége
```

```
Harmadik(A,B):
  B:=''
  Ciklus i=1-től hossz(A)-ig
    ha A(i) mássalhangzó akkor B:=B+'-'+A(i)
                       különben B:=B+A(i)
  Ciklus vége
Eljárás vége
```

```
Negyedik(A,B):
  B:=A(1)
  Ciklus i=2-től hossz(A)-1-ig
    ha A(i) mássalhangzó akkor B:=B+'-'+A(i)
                          különben B:=B+A(i)
  Ciklus vége
  B:=B+A(hossz(A))
Eljárás vége
```

Kilencedik-tizedik osztályosok

1. feladat: Sorszámozás (20 pont)

János gazda kertjében N*N almafát ültetett, négyzetes elrendezésben. A fákat ültetési sorrendben sorszámozta. Adj olyan képletet, amely megadja, hogy az i-edik sorban levő j-edik (a sorokat és oszlopokat 1-től sorszámozzuk) fának mi a János gazda által adott sorszáma, ha a sorszámozás az alábbi:

A.

1	2	3	…	N
N+1	N+2	N+3		2*N
2*N+1				3*N
…				…
				N*N

B.

	…	2*N+1	N+1	1
			N+2	2
			N+3	3
			…	…
N*N	…	3*N	2*N	N

C.

1	3	6	10	…
2	5	9		
4	8			
7				
…				N*N

D.

1	4	9		N*N
2	3	8		
5	6	7		
…				

2. feladat: Gépi nyelv (16 pont)

Egy számítógép az alábbi típusú utasításokat érti, és csak az A és B regisztert használhatja:

Utasítás	Jelentése
ADD A,B	A:=A+B
SUB A,B	A:=A-B
MOV A,B	B:=A
INP A	Beolvasás A-ba
OUT B	Kiírás B-ből

Egy szám kétszeresét kiszámító program ezen a nyelven:

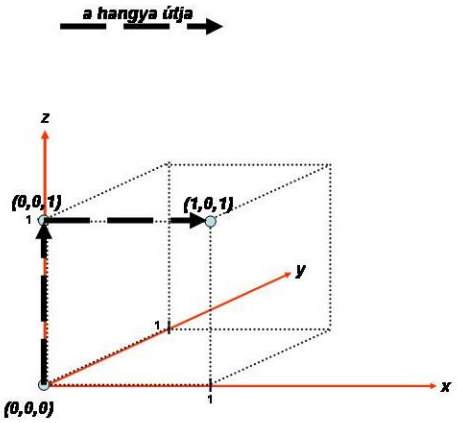
```
INP A
ADD A,A
OUT A
```

Készíts programot ezen a nyelven, ami minimális számú utasítással kiszámolja

A. egy szám 16-szorosát;
B. egy szám 13-szorosát;
C. egy szám 15-szörösét;
D. egy szám 29-szeresét!

3. feladat: Kocka (28 pont)

Egy egységkocka élein (kívül) egy hangya mászik. Kezdetben a (0,0,0) pontban van és a z-tengely irányába néz, azaz biztosan a (0,0,1) pontig fog haladni rajta. Ha egy él végére ér, akkor a két lehetséges továbbhaladási irány közül vagy a jobbra, vagy a balra levő élt választja.



Példa:

A hangya a `JJBJ` döntés sorrend alapján a `(0,0,0),(0,0,1),J,(1,0,1),J,(1,0,0),B,(1,1,0),J,(0,1,0)` pontokon halad keresztül.

Add meg a hangya útját a következő döntéssorozatok esetén!

A. `BBBBBBB` B. `BJBJB` C. `BBJBJJBBJ` D. `JJBBJJB`

4. feladat: Szótagoló (18 pont)

Az alábbi eljárások egyszerű magyar szavakat szótagolnak. Az A szöveges változóban van a szó, a B szöveges változóba teszik a szótagolt szavat. Egyik sem működik helyesen minden magyar szóra. Melyik milyen típusú szavakra működik helyesen? Adj egy-egy példát és fogalmazd meg általánosan is!

```
Első(A,B):
  B:=A(1)
  Ciklus i=2-től hossz(A)-1-ig
    ha A(i) magánhangzó és A(i-1) mássalhangzó akkor  B:=B+A(i)+'-'
                                                különben B:=B+A(i)
  Ciklus vége
  B:=B+A(hossz(A))
Eljárás vége

Második(A,B):
  B:=A(1)
  Ciklus i=2-től hossz(A)-ig
    ha A(i) mássalhangzó és A(i-1) mássalhangzó akkor  B:=B+'-'+A(i)
                                                 különben B:=B+A(i)
  Ciklus vége
Eljárás vége

Harmadik(A,B):
  B:=A(1)
  Ciklus i=2-től hossz(A)-ig
    ha A(i) mássalhangzó és A(i-1) magánhangzó akkor  B:=B+'-'+A(i)
                                                 különben B:=B+A(i)
  Ciklus vége
Eljárás vége

Negyedik(A,B):
  B:=A(1)
  Ciklus i=2-től hossz(A)-ig
    ha A(i) magánhangzó és A(i-1) magánhangzó akkor  B:=B+'-'+A(i)
                                                 különben B:=B+A(i)
  Ciklus vége
Eljárás vége
```

5. feladat: Rácsháló (18 pont)

Egy ágyat úgy készítettek el, hogy a két oldala közé erős rugókat feszítettek ki, majd ezen rugósorokat bizonyos pontokon – szabályos elrendezésben – összekötötték. Egy adott betűvel jelölt tartományból egy hangya indul el egy másik betűvel jelölt tartományba. Add meg, hogy a hangya minimum hány tartományon kell keresztül haladjon (a kezdő- és a végtartományt is beleszámítva), hogy eljusson:

A. Az A tartományból a B tartományba?

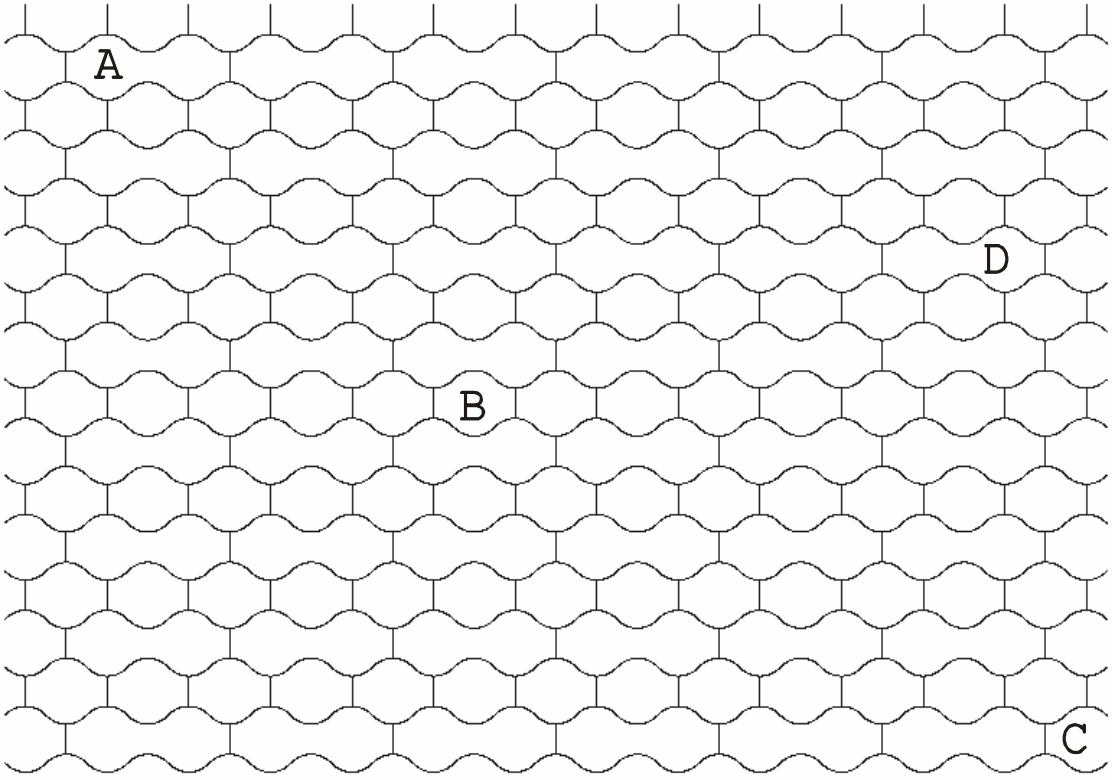
B. Az A tartományból a C tartományba?

C. Az A tartományból a D tartományba?

D. A B tartományból a C tartományba?

E. A B tartományból a D tartományba?

F. A C tartományból a D tartományba?



Tizenegyedik-tizenharmadik osztályosok

1. feladat: Sorszámozás (21 pont)

János gazda kertjében N*N almafát ültetett, négyzetes elrendezésben. A fákat ültetési sorrendben sorszámozta. Adj olyan képletet, amely megadja, hogy az i-edik sorban levő j-edik (a sorokat és oszlopokat 1-től sorszámozzuk) fának mi a János gazda által adott sorszáma, ha a sorszámozás az alábbi:

A.

1	2	3	…	N
N+1	N+2	N+3		2*N
2*N+1				3*N
…				…
				N*N

B.

1	2*N	2*N+1	…	
2		…		
3				
…	…			
N	N+1	3*N	…	

C.

1	3	6	11	…
2	4	8		
5	7	9		
10				
…				N*N

D.

1	4	9		N*N
2	3	8		
5	6	7		
…				

2. feladat: Vonatok (16 pont)

Egy vasútvonal két állomásán egy nap feljegyeztük, az összes, a másik felé áthaladó vonat indulási, illetve érkezési idejét. Feltételezzük, hogy a vonatok a vizsgált szakaszon nem előzhetik meg egymást és egymással szemben soha sem haladhat két vonat. N adatpárt ismerünk, mindegyike egy állomás sorszámot és egy időpontot tartalmaz, idő szerinti sorrendben. Ezek alapján a program találja ki, hogy melyik adatpárban van indulási és melyikben érkezési idő.

A. Mit ír ki az alábbi algoritmus?

B. Mi a `Sor` szerepe az algoritmusban?

C. Mi a szerepe a (*)-gal jelölt sornak?

```
SorInicializálás
Ciklus amíg nem Vége?
  Olvas(állomás,idő)
  Ha Üres?(Sor) akkor típus:=állomás (*)
  Ha állomás=típus akkor  Sorba(idő)
                  különben Sorból(t); Ír(idő-t)
Ciklus vége
```

D. Mi lesz a sor tartalma lépésenként a ciklusmag elején, ha az adatok a következők: (1,1),(1,3),(2,4),(1,4),(1,5),(2,5),(2,6),(2,7),(2,8),(2,9),(1,13),(1,14)?

E. Mi lesz a legkisebb és a legnagyobb kiírt szám a fenti adatokra?

3. feladat: Hálózat (24 pont)

Bergengóciában N város között kell kiépíteni a hálózatot. K héten keresztül kapunk egy-egy újabb árajánlatot két város közötti közvetlen hálózati kapcsolat kiépítésére. Minden hétre – ha lehetsége s– az addig beérkezett javaslatok alapján meg kell adni egy tervet, hogy mely város-párok között építsenek ki közvetlen hálózati kapcsolatot, hogy bármely városból bármely másik elérhető legyen közvetlenül vagy más városokon keresztül, de úgy, hogy az építés összköltsége minimális legyen. A terv összekötendő város-párokból és a hálózat kiépítés teljes költségéből áll.

Példa:

```
N=3, K=4
1: (1-3,100)         NINCS MEGOLDÁS
2: (2-3,50)    (1-3), (2-3)  Költség: 150
3: (1-3,60)    (1-3), (2-3)  Költség: 110
4: (1-2,40)    (1-2), (2-3)  Költség: 90
```

Add meg a minimális számú várospárt és a kiépítés költségét hetenként az alábbi ajánlatok esetén:

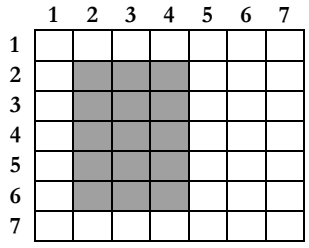
```
A. N=5, K=8          B: N=5,K=10      C. N=5, K=10
   1: (1-2,100)         1: (1-5,100)     1: (5-4,100)
   2: (4-5,100)         2: (1-4,110)     2: (2-3,110)
   3: (2-3,100)         3: (2-3,120)     3: (1-5,120)
```

```
 4: (4-3,100)        4: (1-3,130)     4: (3-4,130)
 5: (1-3,80)         5: (3-5,120)     5: (1-2,100)
 6: (3-5,80)         6: (2-4,110)     6: (2-4,100)
 7: (2-4,80)         7: (3-4,90)      7: (3-5,100)
 8: (5-2,80)         8: (2-5,90)      8: (2-1,80)
                     9: (1-2,100)     9: (4-3,80)
                    10: (4-5,80)     10: (1-3,80)
```

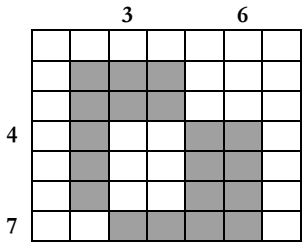
4. feladat: Mobiltelefon (20 pont)

Egy mobiltelefonnak fekete-fehér képernyője van. Különböző méretű képeket kell megjeleníteni XOR műveletekkel. A `XOR(B,F,J,A)` művelet invertálja az (B,F) bal felső és (J,A) jobb alsó sarkú téglalap képpontjait. Kezdetben minden képpont fehér.

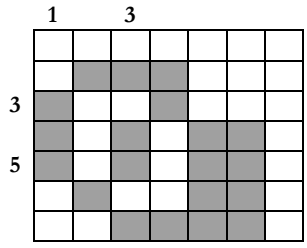
A 3. ábrán levő képet a következő műveletekkel lehet előállítani: A `XOR(2,2,4,6)` művelet hatására az 1. ábrán látható képet kapjuk. Ebből a `XOR(3,4,6,7)` művelet a 2. ábrán látható képet hozza létre. Végül a `XOR(1,3,3,5)` művelet a kívánt képet állítja elő.



1. ábra



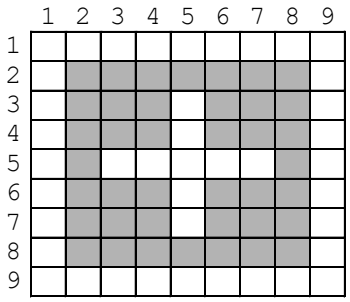
2. ábra



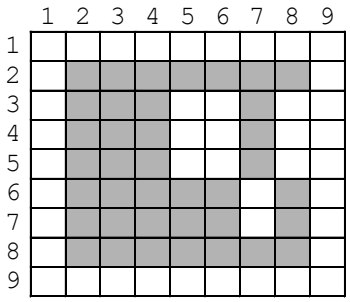
3. ábra

Minden képet az üres képből kiindulva minél kevesebb XOR művelettel kell előállítanod.

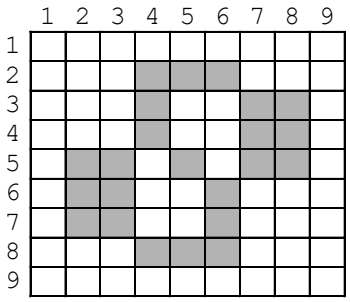
A.



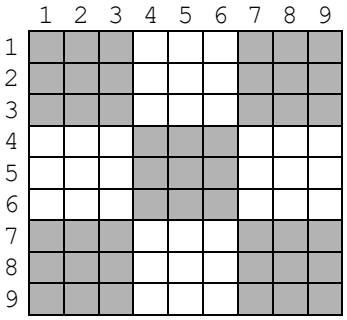
B.



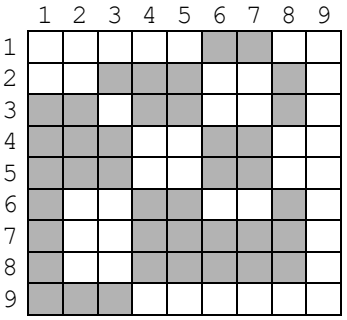
C.



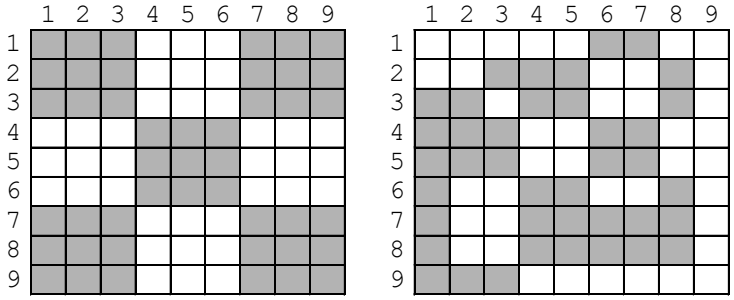
D.



E.



Add meg az egyes ábrákhoz tartozó minimális számú XOR műveletet! Ha több azonos lépésszámú megoldás is lenne, elég egyet megadnod.



5. feladat: Véletlen (19 pont)

A minőségellenőrzési vizsgálatok egyik alapvető módszere, hogy N elkészült termék közük K darabot (K<N) kell kiválasztani konkrét vizsgálatokra. A vizsgálat azonban csak akkor tekinthető hitelesnek, ha az N termék bármelyike azonos eséllyel kerülhet a kiválasztott K termék közé.

Add meg, hogy az alábbi algoritmusok közül melyek teljesítik ezt a feltételt, amelyek nem, azok miért nem? A hibásak közül lehetnek olyanok, amelyek jól működnek, ha N=K+1, melyek ezek? (Mindegyik az N elemű X vektor elemei közül tesz K elemet az Y vektorba. A `véletlen(K)` függvény 1 és K közötti egész, a `véletlenszám` függvény pedig 0 és 1 közötti valós véletlenszámot ad.)

```
A:
  Ciklus i=1-től K-ig
    Y(i):=X(i)
  Ciklus vége
  Ciklus i=K+1-től N-ig
    Y(véletlen(K)):=X(i)
  Ciklus vége
Eljárás vége.
```

```
B:
  Ciklus i=1-től K-ig
    Y(i):=X(i)
  Ciklus vége
  Ciklus i=K+1-től N-ig
    Ha véletlenszám<K/(K+1) akkor Y(véletlen(K)):=X(i)
  Ciklus vége
Eljárás vége.
```

```
C:
  Ciklus i=1-től K-ig
    Y(i):=X(i)
  Ciklus vége
  Ciklus i=K+1-től N-ig
    Ha véletlenszám<K/i akkor Y(véletlen(K)):=X(i)
  Ciklus vége
Eljárás vége.
```

```
D:
  DB:=0
  Ciklus i=1-től N-ig
    Ha véletlenszám<K/N akkor DB:=DB+1; Y(DB):=X(i)
  Ciklus vége
Eljárás vége.
```

```
E:
  DB:=0
  Ciklus i=1-től N-ig
    Ha véletlenszám<(K-DB)/(N-i+1) akkor DB:=DB+1; Y(DB):=X(i)
  Ciklus vége
Eljárás vége.
```

2004. Második forduló

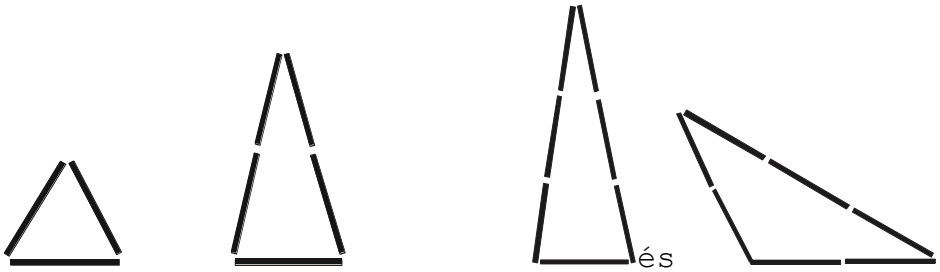
Ötödik-nyolcadik osztályosok

1. feladat: Gyufák (23 pont)

Készíts programot, amely megadja, hogy N gyufaszálból (3≤N≤100) hány különböző háromszöget lehet összerakni!

Példák:

N=3-ra 1-féle N=5-re 1-féle N=7-re 2-féle



2. feladat: Legolcsóbbak (24 pont)

Egy piacon N egymást követő napon árulnak almát. Arra vagyunk kíváncsiak minden napon, hogy az addigi napok közül mely K napon lehetett a legolcsóbban almát venni!

Írj programot, amely beolvassa a napok számát (1≤N≤100), majd K értékét (1≤K≤10), majd ezután egyesével olvassa az egyes napokon az alma árát, s minden beolvasás után kiírja azon K nap sorszámát növekvő sorrendben, amelyeken a legolcsóbban lehetett almát venni! Amíg nem volt K nap, addig a program ne írjon ki semmit!

Példa:

```
N=10, M=4
1. Be: 80
2. Be: 70
3. Be: 75
4. Be: 90     Ki: 1 2 3 4
5. Be: 100  Ki: 1 2 3 4
6. Be: 60     Ki: 1 2 3 6
7. Be: 77     Ki: 2 3 6 7
8. Be: 80     Ki: 2 3 6 7
9. Be: 77     Ki: 2 3 6 7, de a 2 3 6 9 is jó megoldás
10. Be: 90  Ki: 2 3 6 7, de a 2 3 6 9 is jó megoldás
```

3. feladat: Tanév (28 pont)

Bergengóciában a következő szabályok szerint számítják a tanévet:

A. Az első tanítási nap szeptember első hétfője.

B. Az őszi szünet az október 23.-adikát tartalmazó teljes hét, az őt megelőző szombattal és vasárnappal együtt.

C. A téli szünet két teljes hét, úgy, hogy tartalmazza karácsonyt és az újévet is, valamint az azt megelőző szombat-vasárnap.

D. A tavaszi szünet a húsvétot megelőző teljes hét, húsvéthétfő, valamint a hetet megelőző szombat-vasárnap.

E. Az utolsó tanítási nap június második péntekje.

Készíts programot, amely beolvassa, hogy melyik évben vagyunk, szeptember elseje a hét hányadik napjára esik (hétfő az első, vasárnap a hetedik nap), valamint hogy húsvét vasárnap hányadik hónap hányadikán van, majd kiírja a tanév rendjét: első tanítási nap, az egyes szünetek előtti utolsó és szünetek utáni első tanítási napok, valamint az utolsó tanítási nap.

Példa:

```
Év: 2003,
Szeptember 1. a hét hányadik napja: 1,
húsvét vasárnap: 4. hónap 11.
```

Kilencedik-tizedik osztályosok

1. feladat: Másolatok (15 pont)

A könyvnyomtatás kora előtt a könyveket emberek másolták, s másolás során néha hibát vétettek. Emiatt a különböző időben készült másolatok több helyen különbözhetnek egymástól. Ismerjük egy szöveg több változatát, az eredeti szöveget azonban nem.

Írj programot, amely megadja, hogy mi lehetett az eredeti szöveg! Másoláskor legfeljebb K olyan hiba keletkezhetett, hogy a másolt szöveg egy-egy betűjét megváltoztatták. Mivel ugyanazon a helyen kis eséllyel hibáztak a különböző másolók, ezért azt a karaktert vesszük a hibás helyeken helyes karakternek, amely a másolatok többségében fordul elő. Ha ilyen nincs, akkor az adott pozícióra # jelet kell tenni.

A MASOLAT.BE állomány első sorában a szövegek száma ($1 \le N \le 10$), az egy másoláskor keletkező hibák maximális száma ($1 \le K \le 10$) és a szövegek hossza ($1 \le M \le 10\,000$) van. A következő N sorban a másolatok vannak.

A MASOLAT.KI állomány első sorába az előállított eredeti szöveget kell írni!

Példa:

MASOLAT.BE

```
5 3 26
ABCDEFGHIZZLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUAWXYZ
ABCDEFGHIJKLMNOPQRSTUBWXYZ
ABXXXFGHIJKLMNXPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

MASOLAT.KI

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

2. feladat: Alma (12 pont)

Egy piacon M árus N egymást követő napon árul almát. Az árusok különböző napokon kezdenek almát árulni, s ettől kezdve, amíg más árat nem adnak, ugyanazon az áron adják az almát. Arra vagyunk kíváncsiak minden napon, hogy aznap mely K árustól lehet a legolcsóbban almát venni!

Írj programot, amely megadja minden napra, hogy aznap mely K árustól lehet a legolcsóbban almát venni, ha van aznap egyáltalán K árus!

Az ALMA.BE állomány első sorában az árusok száma (1≤M≤100), a napok száma (1≤N≤1000), és a K értéke (1≤K≤M) van. Az állomány további sorai egy-egy árus érkezését írják le. Mindegyik sorban három szám van: az érkezés napja (1≤nap≤N, a sorok eszerint növekvő sorrendben jönnek), az árus sorszáma (1≤sorszám≤M) és az általa árult alma ára attól a naptól kezdve (0<ár≤1000).

Az ALMA.KI állományba pontosan N sort kell írni, az i-edik sorba az i-edik napon legolcsóbb K árus sorszámát, növekvő sorrendben! Ha aznap nem árult almát K árus, akkor a sorba egyetlen 0-t kell kiírni!

Példa:

ALMA.BE	ALMA.KI	magyarázat
6 8 3	0	az első napon nincs 3 árus
1 1 100	2 3 6	
1 2 90	1 3 6	
2 6 80	1 3 6	a negyedik napon nem jött újabb árus
2 3 70	1 3 6	az ötödik napon nem jött újabb árus
2 5 120	1 3 6	a hatodik napon nem jött újabb árus
3 1 60	3 4 6	
3 4 100	3 4 6	a nyolcadik napon nem jött újabb árus
7 1 120		
7 4 75		

3. feladat: Pakolás (15 pont)

Egy vállalat udvarán egyetlen sorban vannak az elszállításra várakozó üres ládák. Három különböző típusú láda van, jelölje ezeket A, B és C. Minden láda egyik oldalán nyitott kocka alakú. Az A-típusú láda a legnagyobb és a C-típusú a legkisebb. Tehát minden C-típusú láda belerakható A-típusú és B-típusú ládába, minden B-típusú belerakható A-típusúba és A-típusúba belerakható B-típusú, majd ebbe egy C-típusú. Az a cél, hogy a ládákat úgy pakoljuk össze, hogy a lehető legkevesebb összepakolt láda legyen. A pakolást olyan robot végzi, amely a ládasor felett tud mozogni mindkét irányban, de ládát csak balról jobbra mozogva tud szállítani.

Írj programot, amely megadja, hogy legkevesebb hány ládába lehet összepakolni a ládasort!

A PAKOL.BE állomány első sorában a ládák száma (1≤N≤10000) van. A második sor pontosan N karaktert tartalmaz (szóközök nélkül), a ládasor leírását. Minden karakter vagy 'A', vagy 'B' vagy 'C'.

A PAKOL.KI állomány első és egyetlen sorába azt a legkisebb K számot kell írni, amelyre a bemeneti ládasor összepakolható K ládába!

Példa:

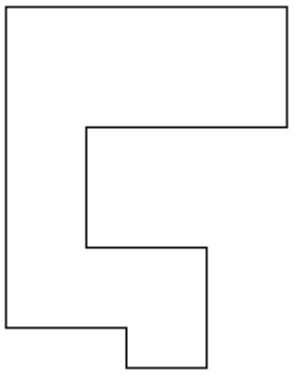
PAKOL.BE	PAKOL.KI
10	6
ABACACBCCA	

4. feladat: Kivágás (15 pont)

János gazda a legelőjének egy részét kerítéssel vette körbe. A kerítést úgy készítette, hogy egy térképen kijelölte a kerítésoszlopok helyét. Minden kerítésoszlopot pozitív egész koordinátájú pontra helyezett, és bármely két, egymást követő kerítésoszlop közötti kerítésszakasz párhuzamos vagy az X, vagy az Y tengellyel. A legelőn sok fa található, amelyek közül néhányat ki akar vágni. A kivágandó fák helyét ismeri, mindegyik helye pozitív egész értékű koordinátáival van megadva. Ki szeretné számítani, hogy a kivágandó fák melyike esik bele az elkerített részbe.

Írj programot, amely megadja minden kivágandó fára, hogy az elkerített részen belül, avagy kívül van-e!

A KIVAG.BE állomány első sorában a kerítésoszlopok száma (4≤N≤10 000) van. A második sor pontosan 2*N darab pozitív egész számot tartalmaz. Az i-edik számpár az i-edik kerítésoszlop X és Y-koordinátájának értéke. Az oszlopokat az óramutató járásával ellentétes körüljárási irányban adjuk meg. A harmadik sor a kivágandó fák számát tartalmazza (2≤M≤100). A további M sor mindegyike egy-egy kivágandó fa X és Y-koordinátáját tartalmazza (1≤X, Y≤20000).



A bemenetre teljesül, hogy bármely kerítésszakasz párhuzamos vagy az X, vagy az Y tengellyel, továbbá bármely két kerítésszakasz nem metszi egymást. Az is teljesül, hogy a kerítésoszlopok helye különbözik a fák helyétől és egyetlen fa sem esik kerítésszakaszra.

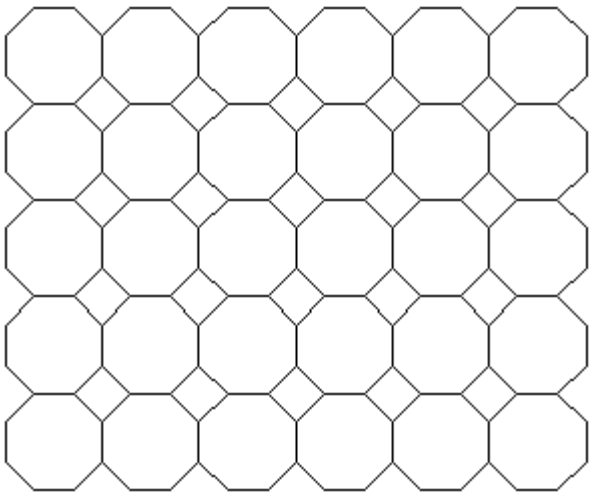
A KIVAG.KI állományba pontosan N sort kell kiírni! Az i-edik sorba az IGEN szót kell írni, ha az i-edik fa az elkerített részen belül van, egyébként a NEM szót!

Példa:

KIVAG.BE	KIVAG.KI
10	NEM
1 2 4 2 4 1 6 1 6 4 3 4 3 7 8 7 8 10 1 10	IGEN
3	IGEN
4 5	
3 8	
7 9	

5. feladat: Rács (18 pont)

Egy üvegrács 100*100 nyolcszög alakú lapból áll, amely a síkot nyolcszögekre és közöttük levő négyzetekre osztja. Az egyik nyolcszögből indulva egy hangya mászik az üveglapokon, adott irányban. Ha egy új lapra ér, akkor új haladási irányt választ magának, s végül biztosan egy nyolcszögben fog megállni.



A hangya útját irányok sorozatával kódolhatjuk. Egy nyolcszögből északra (E), északkeletre (EK), keletre (K), délkeletre (DK), délre (D), délnyugatra (DN), nyugatra (N), illetve északnyugatra (EN) mehet. Négyzetből csak négy irányban távozhat: északkeletre (EK), délkeletre (DK), délnyugatra (DN), illetve északnyugatra (EN).

Írj programot, amely irányokból álló útra megadja, hogy

A. hány négyzeten megy keresztül?

B. hol vagyunk a végén?

C. hány mezőt érint többször?

A RACS.BE állomány első sorában a hangya kezdőpozíciója (1≤KX,KY≤100) van (a bal alsó sarok az (1,1) pozíció, a jobb felső pedig a (100,100)). Az állomány végéig következő sorokban egy-egy irány kódja van: amerre a hangyának tovább kell másznia.

A RACS.KI állományba három sort kell írni, a három kérdésre adott választ! Az első és a harmadik sorban egy-egy egész szám legyen, a második sorban pedig a hangya végső X, illetve Y koordinátája!

Példa:

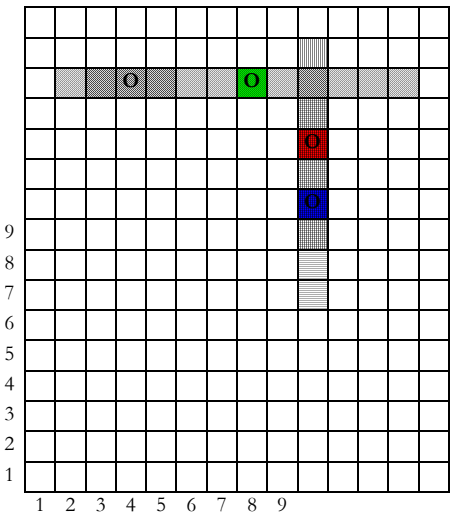
```
RACS.BE         RACS.KI
1 1             4
K               1 1
EK              1
EK
DK              Az út: (1,1), (2,1), négyzet, (3,2), négyzet,
DK                     (4,1), (4,2), négyzet, (3,3), négyzet,
E                      (2,2), (1,2), (1,1)
EN
EN
DN
DN
N
D
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Őr (15 pont)

Egy N*M-es téglalap alakú teremben K őr sétál az őrzött területén, az ábra szerint vagy vízszintes, vagy függőleges irányban. Mindegyik őr L lépésre távolodhat el a kiinduló helyétől, azaz pontosan 2*L+1 mezőt őriz.

Készíts programot, amely megadja, hogy hány helyen találkozhat két őr, melyik melyikkel és mikor! Két őr akkor találkozik, ha egy időegységben ugyanazon a pozíción van!



Az OR.BE állomány első sorában a téglalap sorai (1≤N≤100) és oszlopai (1≤M≤100) száma, valamint az őrök száma (1≤K≤10) van. A következő K sor egy-egy őrt ír le. A sor első karaktere V, ha az őr vízszintesen, illetve F, ha függőlegesen halad. (A vízszintesen haladó őr először balra indul, a függőlegesen haladó pedig felfelé, ha az őrzött terület végére ér, akkor visszafordul.) A betűt követi az őr sorának (1≤S≤N) és oszlopának (1≤O≤M) sorszáma, majd pedig az a távolság (1≤L≤10), amennyire az őr eltávolodhat kezdő helyétől. (Az őr biztosan a téglalapon belül marad.)

Az OR.KI állomány első sorába a találkozási helyek T számát kell írni! A következő T sor mindegyike egy találkozást ír le. Az első két szám a két találkozó őr sorszáma, a következő kettő a találkozás sorának (1≤TS≤N) és oszlopának (1≤TO≤M) sorszáma, az ötödik pedig a kezdőpillanattól az első találkozásig eltelt idő (ha a két őr kezdetben ugyanazon a helyen áll, akkor ez a szám 0)! Minden őr-párra csak az első találkozást szabad megadni! A találkozásokat időrendi sorrendben kell kiírni, az azonos időbeli találkozások sorrendje tetszőleges!

Példa:

```
OR.BE                    OR.KI
100 100 4                2
F 10 10 3                3 4 14 5 3
F 12 10 3                2 3 14 10 14
V 14 8 6
V 14 4 1
```

2. feladat: Szövegek (12 pont)

A könyvnyomtatás kora előtt a könyveket emberek másolták, s másolás során néha hibát vétettek. Emiatt a különböző időben, különböző könyvpéldányokról készült másolatok több helyen különbözhetnek egymástól. Ismerjük egy szöveg több változatát, keletkezési idejük sorrendjében.

Írj programot, amely megadja, hogy melyik melyikből keletkezhetett! Másoláskor legfeljebb K olyan hiba keletkezhetett, hogy a másolt szöveg valahány betűjét megváltoztatták. Feltehető, hogy egy megváltozott betű tovább már nem változott. Azt tartjuk egy másolt szöveg eredetijének, amelytől a fenti feltételek mellett a legkevesebb betűben tér el. (Ha több ilyen is van, akkor bármelyikből keletkezhetett.)

A MASOL.BE állomány első sorában a szövegek száma ($1 \le N \le 10$) és az egy másoláskor keletkező hibák maximális száma ($1 \le K \le 10$) van. A második sorban található az eredeti szöveg (legfeljebb 1000 karakteres), majd az ezt követő N-1 sorban a másolatok keletkezési sorrendben.

A MASOL.KI állomány első sorába pontosan N-1 számot kell írni! Az i-edik szám annak a szövegnek a sorszáma, amelyből az i+1-edik szöveg keletkezett!

Példa:

```
MASOL.BE                          MASOL.KI
4 3                               1 1 2
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNABCRSTUVWXYZ
ABCDEFGHIJKLMNOBCDSTUVWXYZ
ABCDEFGHIJKLMNABCDSTUVWTYZ
```

3. feladat: Hálózati központ (15 pont)

Minden számítógépes gerinchálózat csomópontokból, és bizonyos csomópont-párokat összekötő, kétirányú közvetlen adatátvitelt biztosító kommunikációs vonalakból épül fel. Adott U és V csomópont közötti útvonalon olyan $P_0, P_1, \ldots, P_k$ csomópontsorozatot értünk, ahol $U=P_0$, $V=P_k$ és minden i-re a P_i és a P_{i+1} ($i=0,\ldots k-1$) csomópontpárt közvetlen vonal köti össze. Ekkor ennek az útvonalnak a hossza k. A feladatban szereplő hálózatról tudjuk, hogy bármely két csomópontja között pontosan egy útvonal létezik. Minden U csomópontra fontos jellemző adat az U-ból induló leghosszabb útvonal hossza, jelöljük ezt az értéket Táv(U)-val. A hálózat olyan U csomópontját, amelyre a Táv(U) a legkisebb, a hálózat központjának nevezzük. Például, az ábrán látható hálózatban a 7-es és a 10-es csomópont központ.

Írj programot, amely meghatározza adott hálózat egyik központját!

A KOZPONT.BE állomány első sorában a csomópontok száma van ($2 \le N \le 1000$). A csomópontokat az $1,\ldots,N$ számokkal jelöljük. A következő N sor adja meg a csomópontok közötti közvetlen kapcsolatokat. Az állomány i+1-edik sorában azok a csomópontok vannak felsorolva, amelyeket közvetlen vonal köti össze az i-edik csomóponttal. A sort a 0 szám zárja, ami nem csomópontsorszám.

A KOZPONT.KI állomány első és egyetlen sorába a hálózat egy központjának sorszámát kell írni!

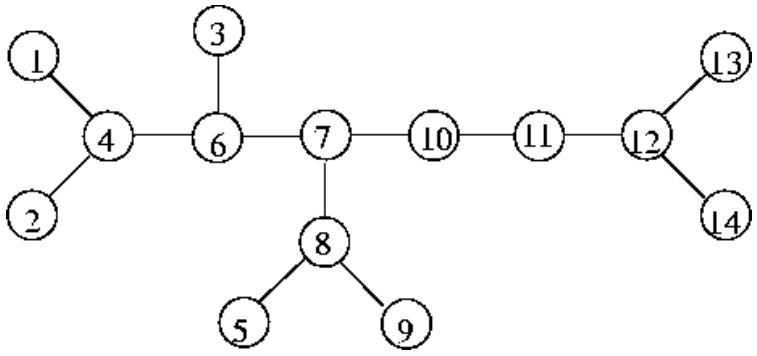
Példa:

KOZPONT.BE

```
14
4 0
4 0
6 0
1 6 2 0
8 0
4 3 7 0
6 10 8 0
7 5 9 0
8 0
7 11 0
10 12 0
11 13 14 0
12 0
12 0
```

KOZPONT.KI

```
7
```



4. feladat: Fényképezés (15 pont)

Egy rendezvényre N vendéget hívtak meg. Minden vendég előre jelezte, hogy mettől meddig lesz jelen. A szervezők fényképeken akarják megörökíteni a rendezvényen résztvevőket. Azt tervezik, hogy kiválasztanak K időpontot és minden kiválasztott időpontban az akkor éppen jelenlevőkről csoportképet készítenek. Az a céljuk, hogy a lehető legkevesebb képet kelljen készíteni, de mindenki rajta legyen legalább egy képen.

Írj programot, amely kiszámítja, hogy legkevesebb hány fényképet kell készíteni, és megadja azokat az időpontokat is amikor csoportképet kell készíteni!

A FENYKEP.BE állomány első sorában a vendégek száma van ($1 \le N \le 3000$). A következő N sor mindegyike egy vendég E érkezési és T távozási időpontját tartalmazza ($1 \le E < T \le 1000$). Ha egy fényképet az x időpontban készítik és $E \le x < T$, akkor azon a fényképen rajta lesz az E időben érkező és T időben távozó vendég.

A FENYKEP.KI állomány első sorába a készítendő fényképek K számát kell írni! A második sor pontosan K egész számot tartalmazzon, azon időpontokat (tetszőleges sorrendben), amikor a csoportképeket készíteni kell!

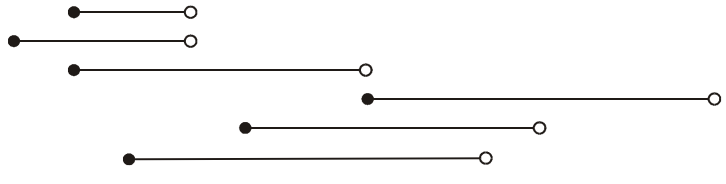
Példa:

FENYKEP.BE

```
6
2 4
1 4
2 7
7 13
5 10
3 9
```

FENYKEP.KI

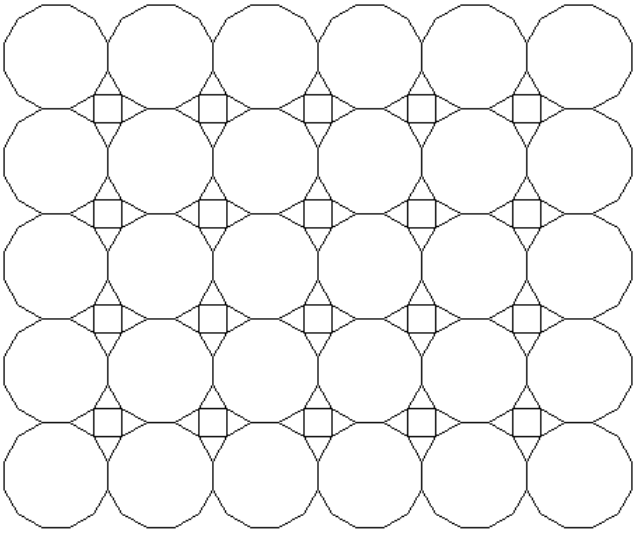
```
2
3 9
```



5. feladat: Rácsháló (18 pont)

Egy üvegrács 100*100 tizenkétszög alakú lapból áll, amely a síkot tizenkétszögekre és közöttük levő háromszögekre és négyzetekre osztja. Az egyik tizenkétszögből indulva egy hangya mászik az üveglapokon, adott irányban. Ha egy új lapra ér, akkor új haladási irányt választ magának, s végül biztosan egy újabb tizenkétszögben fog megállni.

A hangya útját irányok sorozatával kódolhatjuk. Egy tizenkétszögből északra (E), észak-északkeletre (EEK), kelet-északkeletre (KEK), keletre (K), kelet-délkeletre (KDK), dél-délkeletre (DDK), délre (D), dél-délnyugatra (DDN), nyugat-délnyugatra (NDN), nyugatra (N), nyugat-északnyugatra (NEN), illetve észak-északnyugatra (EEN) mehet. A négyzetekből csak négy irányban távozhat: északra (E), keletre (K), délre (D), illetve nyugatra (N). A háromszögekből három irányba léphet, állásuktól függően a tizenkétszögből kilépés irányai közül a megfelelő háromba.



Írj programot, amely irányokból álló útra megadja, hogy

A. hány négyzeten megy keresztül?
B. hol vagyunk a végén?
C. hány mezőt érint többször?

A HALO.BE állomány első sorában a hangya kezdőpozíciója ($1 \le KX, KY \le 100$) van (a bal alsó sarok az (1,1) pozíció, a jobb felső pedig a (100,100)). Az állomány végéig következő sorokban egy-egy irány kódja van: amerre a hangyának tovább kell másznia.

A HALO.KI állományba három sort kell írni, a három kérdésre adott választ! Az első és a harmadik sorban egy-egy egész szám legyen, a második sorban pedig a hangya végső X, illetve Y koordinátája!

Példa:

```
HALO.BE         HALO.KI
1 1             1
E               2 2
K               1
EEK
EEN             Az út: (1,1), (1,2), (2,2), háromszög, (2,3),
KDK                    háromszög, (3,3), háromszög, négyzet,
KEK                    háromszög, (2,2)
DDN
N
D
NDN
```

2004. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Automata (27 pont)

Egy csokoládé árusító automatát úgy alakítottak ki, hogy mindenféle pénzérmét elfogad, de egy vásárláskor maximum 3 érmét dobhatunk be. Ha többféleképen is fizethetünk, akkor a célunk, hogy minél kevesebb érmét használjunk el.

Készíts programot, amely beolvassa egy kiválasztott csokoládé árát ($1 \le AR \le 300$), a pénzérméink számát ($1 \le N \le 100$), majd az N darab pénzérme értékét ($1 \le E[i] \le 100$)!

Ezután a program írja ki, ha lehetséges, hogy a vásárláshoz milyen pénzérméket használhatunk a feltételeknek megfelelően! Ha nem lehetséges legfeljebb 3 érmével fizetni, akkor a program a NEM LEHET szöveget írja ki!

Példa:

```
AR=150, N=4, E=[50,50,100,50] esetén az eredmény: 50+100
```

2. feladat: Szólánc (22 pont)

A szólánc játékot úgy játsszák, hogy valaki mond egy szót, a következőnek a szó utolsó betűjével kezdődő szót kell mondania, az őt követőnek a második szó utolsó betűjével kezdődőt, … és így tovább. A szólánc szavait azonban valaki összekeverte, de tudjuk, hogy ezek a szavak csak egyféle sorrendben rakhatók össze, ezért a sorrendjük biztosan helyreállítható.

Készíts programot, amely beolvassa a szólánc szavai számát (1≤N≤100), majd az N darab összekevert szót (mindegyik legfeljebb 20 karakteres), s ezek alapján kiírja a szóláncot a helyes sorrendben!

Példa:

```
N=4, szavak=AGÁR,KÖRTE,MÁLNA,RETEK
Szólánc= MÁLNA → AGÁR → RETEK → KÖRTE
```

3. feladat: Idő (26 pont)

Az időt (óra, perc, másodperc, századmásodperc) formában tároljuk. Olyan típusú kérdéseket szeretnénk feltenni, hogy ha most 8 óra 10 perc 12 másodperc 3 századmásodperc van, akkor mennyi lesz az idő 80 másodperc múlva, illetve hogy mennyi volt az idő ezelőtt 2 óra 15 perccel. Azaz az idő típusú adatokra el kell készíteni az összeadás és a kivonás műveletet.

Készíts programot, amely beolvas egy időpontot: O, P, MP, SZMP formában (0≤O≤23, 0≤P≤59, 0≤MP≤59, 0≤SZMP≤99), majd egy idő távolságot (0≤A, 0≤B, 0≤C, 0≤D) és kiírja, hogy mennyi volt az idő A óra B perc C másodperc D századmásodperccel korábban, illetve később!

Példa:

```
O=8, P=10, MP=12, SZMP=93   A=0, B=80, C=20, SZMP=10
Előtte: 6 óra 49 perc 52 másodperc 83 századmásodperc
Utána: 9 óra 30 perc 33 másodperc 3 századmásodperc
```

Kilencedik-tizedik osztályosok

1. feladat: Rémhír (15 pont)

Rémhírek úgy keletkeznek, hogy egy vagy több ember elmondja néhány másik embernek, azok a rémhírt továbbadják, … és így tovább.

Készíts programot, amely az emberek kikérdezése alapján megadja azokat,

A. akiktől a rémhír elindult (akik nem kapnak hírt másoktól);

B. akik nem adnak tovább a rémhírt;

C. valamint azokat, akik a legtöbb embernek adják tovább a rémhírt!

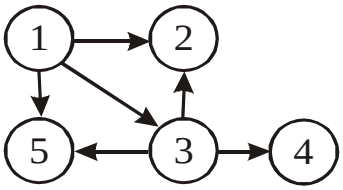
A REMHIR.BE állomány első sorában az emberek száma (1≤N≤1000) és a hírtovábbadások száma (1≤M≤10 000) van. A következő M sor mindegyikében két ember sorszáma van (1≤A≠B≤N), amelynek jelentése: B a rémhírt A-tól hallotta.

A REMHIR.KI állományba három sort kell írni, az elsőbe az A, a másodikba a B, a harmadikba pedig a C részfeladat megoldását! Mindegyik sor az összes megfelelő ember sorszámát tartalmazza,

a soron belül növekvő sorrendben! Ha valamelyik részfeladatra nincs megoldás, az üres sort akkor is ki kell írni!

Példa:

```
REMHIR.BE          REMHIR.KI
10 6               1
1 2                2 4 5
1 3                1 3
3 4
3 2
3 5
1 5
```



2. feladat: Hangya (13 pont)

Egy egységkocka élein egy hangya mászik. Kezdetben a (0,0,0) pontban van és a z-tengely irányába néz, azaz biztosan a (0,0,1) pontig fog haladni rajta. Ha egy él végére ér, akkor a két lehetséges továbbhaladási irány közül vagy a jobbra, vagy a balra levő élt választja.

Írj programot, amely az irányok sorozata alapján megadja a hangya útját!

A HANGYA.BE állomány első sorában a továbbhaladási irányok száma van ($0 \le N \le 1000$). A második sorban a haladási irányok vannak: pontosan N betű, a következő él végén a hangya J betű hatására jobbra, a B betű hatására pedig balra fordul.

A HANGYA.KI állományba N+2 sort kell írni, a hangya pozícióit az út során! Minden sor három számot tartalmazzon, a hangya x-, y-, illetve z-koordinátáját!

Példa:

A hangya a `JJBJ` döntés sorrend alapján a `(0,0,0),(0,0,1),J,(1,0,1),J,(1,0,0),B,(1,1,0),J,(0,1,0)` pontokon halad keresztül.

```
HANGYA.BE          HANGYA.KI
4                  0 0 0
JJBJ               0 0 1
                   1 0 1
                   1 0 0
                   1 1 0
                   0 1 0
```

3. feladat: Azonosító (15 pont)

A processzorgyártó cégek megállapodtak abban, hogy milyen rendszert alkalmaznak az általuk gyártott processzorok azonosítására. Minden cég kap egy betűkészletet, és ezekből kell az azonosító kódot képeznie úgy, hogy minden betű pontosan egyszer szerepeljen az azonosítóban. Például egy cég azt kapta, hogy minden azonosítója egy-egy 'a', 'b', 'c', 'd' és 'x' betűt tartalmazzon. A processzorok a gyártási sorrendben ábécé szerint növekvően kapják mindig a következő azonosítót.

Készíts programot, amely adott azonosítóra kiszámítja az ábécé sorrendben rákövetkező szabályos azonosító!

Az AZON.BE állomány első és egyetlen sorában egy szabályos azonosító van, amely csak kisbetűket tartalmaz az 'a' .. 'z' tartományból és legfeljebb 12 betűből áll.

Az AZON.KI állomány első sorába a rákövetkező szabályos azonosítót kell írni! Ha nincs ilyen, akkor a NINCS szót!

Példa:

AZON.BE	AZON.KI
bdaxc	bdcax

4. feladat: Rejtvény (15 pont)

Egy rejtvény fajtában a betűket egy négyzetes mátrixban helyezik el, s ebben vízszintesen, függőlegesen vagy átlósan bizonyos szavak olvashatók ki (bármilyen irányban haladva).

Készíts programot, amely egy rejtvényben megszámolja, hogy bizonyos szavak hányszor fordulnak elő!

A REJTVENY.BE állomány első sorában a mátrix mérete (1≤N≤100) és a keresett szavak száma (1≤M≤10) van. A következő N sor mindegyikében a mátrix egy-egy sorának betűi vannak, mindegyikben pontosan N darab karakter. Az utolsó M sor mindegyikébe egy-egy keresett szó van.

A REJTVENY.KI állományba pontosan M sort kell írni! Az i-edik sorba az i-edik keresett szó mátrixbeli előfordulásai számát kell írni!

Példa:

REJTVENY.BE	REJTVENY.KI
6 3	1
QWERTZ	2
UIOPAS	2
EGERTL	
LYDCAB	
QZYZTX	
ÓERRÓV	
EGER	
TATA	
ÓZD	

5. feladat: Ütemezés (17 pont)

Adott N darab program, amelyeket egy processzoron kellene végrehajtani. Ismerjük mindegyik program végrehajtásához szükséges időt és a határidejét, ameddig a program végrehajtását be kell fejezni. Kiválasztandó a programoknak egy olyan legnagyobb elemszámú részhalmaza, amelyek végrehajtását lehet úgy ütemezni, hogy minden kiválasztott program végrehajtása befejeződjék a határidejéig.

Írj programot, amely meghatározza a programok egy lehető legnagyobb elemszámú részhalmazát úgy, hogy az összes kiválasztott program végrehajtását lehet úgy ütemezni, hogy minden kiválasztott program végrehajtása befejeződjék a határidejéig! A program adjon is meg egy alkalmas ütemezést!

Az UTEMEZ.BE állomány első sora a programok számát (1≤N≤10 000) tartalmazza. A következő N sor mindegyike két pozitív egész számot tartalmaz, egy program V végrehajtási idejét, illetve H határidejét (1≤V≤H≤10 000).

Az UTEMEZ.KI állomány első sorában a kiválasztott programok M száma legyen! A második sorba M számot, a kiválasztott programok sorszámát kell írni olyan sorrendben, amely megfelel egy határidőket betartó ütemezésnek! Programok egy $p_1, \ldots, p_M$ felsorolása határidőt betartó ütemezés, ha minden i-re (1≤i≤M) az első i program végrehajtási idejének összege nem nagyobb a p_i program határidejénél. Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni!

Példa:

```
UTEMEZ.BE              UTEMEZ.KI
6                      3
4 4                    2 5 3
3 8
3 10
4 9
2 9
4 11
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Hírlánc (15 pont)

Hírek úgy keletkeznek, hogy egy vagy több ember elmondja néhány másik embernek, azok azt tartják igaznak, amit több embertől hallottak, majd azt a hírt adják tovább, … és így tovább. Mindenki azt az egy hírt fogadja el igaznak, amelyet több embertől hallott, mint bármely más hírt.

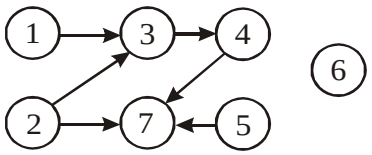
Készíts programot, amely az emberek kikérdezése alapján megadja, hogy melyik milyen hírt tart igaznak! Feltehető, hogy hír nem jár körbe.

A HIRLANC.BE állomány első sorában az emberek száma (1≤N≤200), a hírtovábbadások száma (1≤M≤10 000), valamint a hírt kitalálók száma (1≤K≤N) van. A következő M sor mindegyikében két ember sorszáma van (1≤A≠B≤N), amelynek jelentése: A azt a hírt, amit igaznak tart, B-nek továbbítja. Ha A nem tart igaznak egy hírt sem, akkor nem továbbít semmit B-nek. A további K sor mindegyikében a hírt kitaláló ember sorszáma (1≤sorszám≤N) és a hír tartalma (egyetlen, legfeljebb 10 karakteres szó) van. Mind a K hír különböző. Hír kitalálója nem kaphat hírt másoktól. Minden ember akkor dönt arról, hogy melyik hírt fogadja el igaznak, amikor mindenkitől, aki neki hírt továbbíthat, megkapta a hírt.

A HIRLANC.KI állományba pontosan N sort kell írni! Az i-edik sorban az a hír (legfeljebb 10 karakteres szó) legyen, amit az i-edik ember igaznak tart! Ha valamelyik ember nem tudja, mi az igaz hír, akkor a megfelelő sorba a NINCS szót kell kiírni!

Példa:

```
HIRLANC.BE             HIRLANC.KI
7 6 2                  árvíz
2 3                    földrengés
1 3                    NINCS
3 4                    NINCS
2 7                    NINCS
4 5                    NINCS
4 7                    földrengés
1 árvíz
2 földrengés
```



2. feladat: Azonosító (15 pont)

A processzorgyártó cégek megállapodtak abban, hogy milyen rendszert alkalmaznak az általuk gyártott processzorok azonosítására. Minden cég kap egy betűkészletet, és ezekből kell az azonosító kódot képeznie úgy, hogy minden betű pontosan egyszer szerepeljen az azonosítóban. Például egy cég azt kapta, hogy minden azonosítója egy-egy 'a', 'b', 'c', 'd' és 'x' betűt tartalmazzon. A processzorok a gyártási sorrendben ábécé szerint növekvően kapják mindig a következő azonosítót.

Készíts programot, amely adott azonosítóra kiszámítja a következő két kérdésre a választ.

A. Hányadik a gyártási sorrendben a processzor? (0-tól sorszámozva)

B. Mi az ábécé sorrendben rákövetkező szabályos azonosító?

Az AZON.BE állomány első és egyetlen sorában egy szabályos azonosító van, amely csak kisbetűket tartalmaz az 'a' .. 'z' tartományból és legfeljebb 12 betűből áll.

Az AZON.KI állomány első sorába az A kérdésre adandó választ kell írni! A második sorba a B kérdésre adandó választ kell írni, tehát a sorrendben következő szabályos azonosítót, ha nincs ilyen, akkor a NINCS szót!

Példa:

AZON.BE	AZON.KI
bdaxc	37 bdcax

3. feladat: Ültetés (15 pont)

Az osztályod ünnepségre készül, ahol minden osztály külön széksort kapott, pontosan annyi székkel, amennyi az osztálylétszám. Az osztályfőnök mindenkitől megkérdezte, hogy nevezzen meg egy tanulót aki mellett ülni szeretne.

Készíts programot, amely kiszámít egy olyan ültetési sorrendet, amely a lehető legtöbb tanuló kérését teljesíti!

Az ULTET.BE állomány első sorában a tanulók száma ($1 \le N \le 100$) van. A tanulókat az 1,…,N számokkal azonosítjuk. A második sor pontosan N számot tartalmaz. Az i-edik szám (nem egyelő i-vel) annak a tanulónak a sorszáma (1 és N közötti érték), aki mellett az i-edik tanuló ülni szeretne.

Az ULTET.KI állomány első és egyetlen sorába N számot kell írni! Az i-edik szám annak a széknek a sorszáma legyen, ahova az i-edik tanulót ültetjük a legtöbb kérést kielégítő ültetésben! (Több megoldás esetén bármelyik megadható.)

Példa:

ULTET.BE	ULTET.KI
11 8 1 1 1 6 8 6 9 10 8 10	2 1 3 4 5 6 7 11 10 9 8

4. feladat: Szójáték (15 pont)

A szójáték népszerű játék, sok változata ismert. Tekintsük azt a változatát, amelyben egy adott szövegben elrejtett szót kell megtalálni! Például, az „abrakadabra" szövegben kell megtalálni az „akar" szót. Ennek egy megoldása: „abr***aka***dab***r***a". Hogy érdekesebb legyen a játék, a szövegnek egy olyan legrövidebb részét kell megadni, amelyben még előfordul a keresendő szó (ha van megoldás).

Írj programot, amely meghatározza adott szövegben azt a legrövidebb részt, amely tartalmazza a megadott szót!

A SZOJATEK.BE állomány első sora a keresendő szó hosszát tartalmazza ($1 \le M \le 100$). Az állomány második sora pontosan M karaktert tartalmaz, a keresendő szót. A harmadik sor a szöveg hosszát ($1 \le N \le 300\,000$) tartalmazza. A negyedik sor pontosan N karaktert tartalmaz, a szöveget. A szöveg és a szó is tetszőleges alfabetikus karaktert tartalmazhat.

A SZOJATEK.KI állomány első és egyetlen sorába két egész számot kell írni, I-t és H-t! A szöveg I-edik karakterétől vett H hosszú rész tartalmazza a megadott szót és nincs a szövegnek H-nál

rövidebb része, amely tartalmazná a szót! Ha több ilyen I index lenne, akkor a legkisebbet kell megadni! Ha nincs megoldás, akkor a 0 0 számpárt kell kiírni!

Példa:

SZOJATEK.BE	SZOJATEK.KI
4	4 7
akar	
11	
abr**aka**dab**r**a	

5. feladat: Körutazás (15 pont)

Osztályod azt tervezi, hogy az osztálykirándulás során az egyik napon a környékbeli városok között egy körutazást tesz autóbusszal. Az osztály még nem döntött az útvonalról, csak azt határozta meg, hogy olyan útvonalat kell választani, amely legalább két várost érint a kiindulásin kívül, és hogy az útvonal különböző városokon át haladjon. Az idő rövidsége miatt azt is ki kellett kötni, hogy a lehető legkevesebb várost érintsen a túra. A feladat megoldásához van olyan menetrend, amely tartalmazza, hogy mely városok között van közvetlen kétirányú buszjárat.

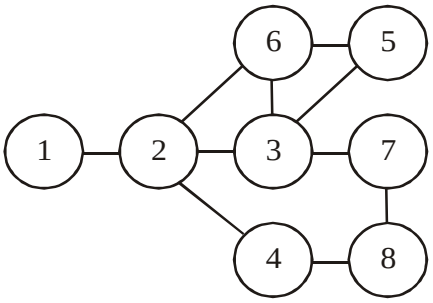
Írj programot, amely meghatároz egy, a követelményeket kielégítő útvonalat!

A KORUT.BE állomány első sora tartalmazza a városok számát (1≤N≤200), a buszjáratok számát (1≤M≤10 000) és a kiindulási város azonosítóját. Az állomány következő M sora egy-egy számpárt (1≤U≠V≤N). tartalmaz. Ez azt jelenti, hogy az U és V város között van közvetlen, mindkét irányban közlekedő buszjárat. Bármely két városra legfeljebb egy buszjárat van.

A KORUT.KI állomány első sorába a körút során érintett városok L számát kell írni! A második sor tartalmazza a követelményeknek megfelelő körút leírását, azaz L város azonosítóját! Az első szám a kiindulási város K azonosítója legyen! Ha nincs megoldás, akkor az állomány első és egyetlen sora a 0 számot tartalmazza!

Példa:

KORUT.BE	KORUT.KI
8 10 2	3
1 2	2 3 6
2 3	
2 4	
3 5	
5 6	
6 2	
6 3	
3 7	
7 8	
4 8	



A verseny végeredménye:

I. korcsoport

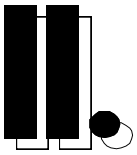
1. Nagy Gergely	Fazekas Mihály Gimnázium, Budapest
Szárnyas Gábor	Bólyai János Általános Iskola és Gimnázium, Szombathely
3. Hegedűs Tamás	Fazekas utcai Általános Iskola, Miskolc
Korándi Dániel	Fazekas Mihály Gimnázium, Budapest
5. Tóth László Márton	Fazekas Mihály Gimnázium, Debrecen
Varga Iván	Bólyai János Általános Iskola és Gimnázium, Szombathely
7. Nagy Krisztián	Berzsenyi Dániel Gimnázium, Budapest
8. Eisenberger András	Fazekas Mihály Gimnázium, Budapest
Radnai Ágnes	Veres Péter Gimnázium, Budapest
10. Csóka Győző	Fazekas Mihály Gimnázium, Debrecen
Peregi Tamás	Berzsenyi Dániel Gimnázium, Budapest

II. korcsoport

1. Vincze János	Fazekas Mihály Gimnázium, Debrecen
2. Incze Attila	Radnóti Miklós Gimnázium, Szeged
3. Kormányos Balázs	Radnóti Miklós Gimnázium, Szeged
Jobbágy László	Árpád Vezér Gimnázium, Sárospatak
5. Acsai Péter	Arany János Református Gimnázium, Nagykőrös
6. Hegedűs Tibor	Földes Ferenc Gimnázium, Miskolc
7. Nikházy László	Kazinczy Ferenc Gimnázium és Szakközépiskola, Győr
Szabó Gábor	Neumann János Középiskola és Kollégium, Eger
9. Soltész Zoltán	ELTE Apáczai Csere János Gimnázium, Budapest
10. Cséri Tamás	Zrínyi Miklós Gimnázium, Zalaegerszeg

III. korcsoport

1. Fehér Gábor	Berzsenyi Dániel Gimnázium, Budapest
2. Rácz Béla András	Fazekas Mihály Gimnázium, Budapest
Isza Péter	Tóth Árpád Gimnázium, Debrecen
4. Konfár András	Radnóti Miklós Gimnázium, Szeged
5. Gruber László	Mechwart András Gépipari és Informatikai SzKI, Debrecen
Sztupák Szilárd Zsolt	Herman Ottó Gimnázium, Miskolc
7. Kaposi Ambrus	Bencés Gimnázium, Pannonhalma
Kocsis István	Földes Ferenc Gimnázium, Miskolc
Ludányi Ákos	Neumann János Középiskola és Kollégium, Eger
Pósfai Márton	Ságvári Endre Gimnázium, Szeged



Nemes Tihamér
Nemzetközi Informatikai Tanulmányi Verseny

2000. Első forduló

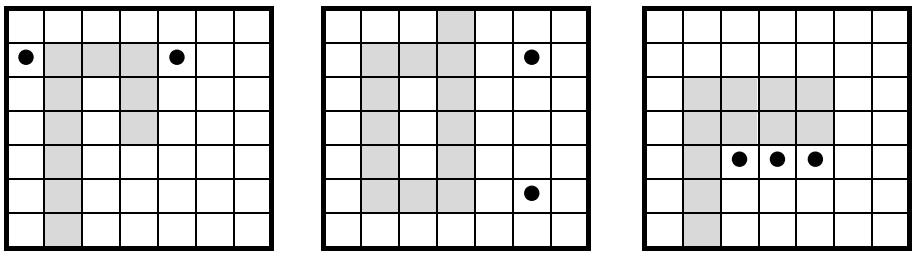
Ötödik-nyolcadik osztályosok

1. feladat: Karesz a robot (30 pont)

A1. a 9. lépés után (a 10.-nél lelépne) B1. a 15. lépés után (a 16.-nál lelépne) C1. a 13. lépés után (a 14.-nél lelépne) 4+4+4 pont

A2. 2 kő B2. 4 kő C2. 6 kő 2+2+2 pont

A3. **B3.** **C3.**



4+4+4 pont

2. feladat: Falfestés (24 pont)

Aa1. 2 lépés 2 pont

Aa2. 2 lépés 2 pont

Aa3. 3 lépés 2 pont

Az egyes menetek utáni vastagságok (helyes számcsoportonként adható pont)

Ab1. 5,3,3,3,3,3, 5,5,5,5,5,5 2+2 pont

Ab2. 1,2,3,5,3,3, 1,2,3,5,5,5 2+2 pont

Ab3. 1,5,4,7,4, 1,5,5,7,6, 1,5,5,7,7 2+2+2 pont

B. Kezdetben az első hely legyen az egyik legvastagabb 4 pont

3. feladat: Válassz ki kettőt (26 pont)

A. `Első` - A=a legnagyobb elem értéke, 3 pont
B= a legkisebb elem értéke, ami előtt volt nála nagyobb vagy egyenlő 3+2 pont

`Második` - A= a legnagyobb elem értéke, 2 pont
B= a második legnagyobb elem értéke, lehet A-val egyenlő 3 pont

`Harmadik` - A= a legnagyobb elem értéke, 2 pont
B= a második legnagyobb elem értéke, A-nál biztos kisebb 3 pont

B. `Első`: ha a számok egyre nagyobbak lesznek 1+3 pont

`Harmadik`: ha a számok egyenlők 1+3 pont

Ha a `Második` is szerepel, az 1 pont levonás.

4. feladat: Ládapakoló robot (20 pont)

A. Első: 1. polc: 5,4,1 3 pont

2. polc: 8,3 2 pont

3. polc: 9,6 2 pont

Második: 1. polc: 9,6 2 pont

2. polc: 8,3 2 pont

3. polc: 5,4,1 3 pont

B. Ha az összes polcon nála kisebb van legfelül. 3 pont

(elég az is, hogy a legelső polcon nála kisebb van legfelül)

C. 6-nál nagyobb legyen 3 pont

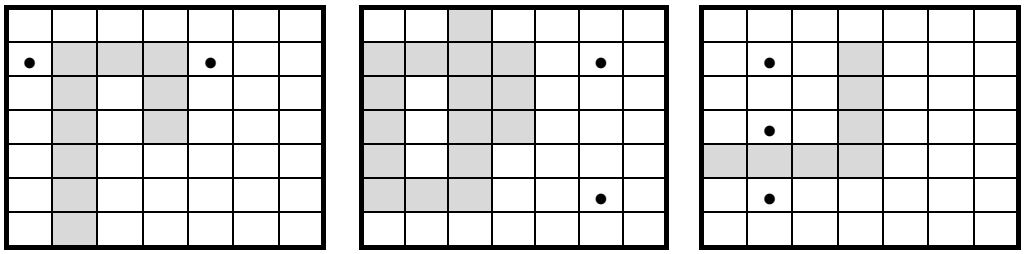
Kilencedik-tizedik osztályosok

1. feladat: Karesz a robot (21 pont)

A1. a 9. lépés után (a 10.-re már lelépne) B1. a 16. lépés után (a 17.-re már lelépne) C1. a 8. lépés után (a 9.-re már lelépne) 3+3+3 pont

A2. 2 kő B2. 8 kő C2. 5 kő 1+1+1 pont

A3. **B3.** **C3.**



Jó az útvonal 3+3+3 pont

2. feladat: Kiszámolós (20 pont)

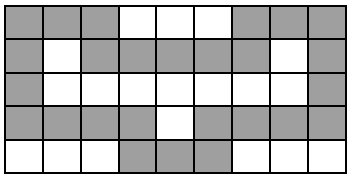
A. Első: 5,10,2, 7,12, 4,9,1,6, 11,3,8,13 2+1+1+1 pont

Második: 5,10,2, 8,1,9, 4,13,12, 3,7,11,6 2+1+1+1 pont

Harmadik: 5,1,11, 9,8,7,6, 4,3,2, 13,12,10 2+1+1+1 pont

B. Az első kerülhet végtelen ciklusba, ha N és K nem relatív prím 2+3 pont

3. feladat: Grafikus utasítások (18 pont)

A.  6 pont

B.

B1. Az eredeti rajz X (vízszintes, kelet-nyugati) tengelyre vett tükörképét rajzolja 6 pont

B2. Az eredeti rajzot 90 fokkal jobbra (az óramutató járásával megegyező irányba) elforgatva rajzolja 6 pont

4. feladat: Rendezőpályaudvar (21 pont)

A. Az érkező szerelvényt bárhol kettévágva igaz legyen, hogy olyan állomás, amire az első és a második részből is kell vagonokat küldeni, legfeljebb K van 5 pont

B. Az érkező szerelvényt bárhol kettévágva, az olyan állomások maximális száma, amire az első és a második részből is kell vagonokat küldeni 5 pont

C1. 3 sínpár 1 pont

1: 1,2,4,6,81 pont

2: 3,9,11 1 pont

3: 5,7,10,12 2 pont

C2. 2 sínpár 2 pont

1: 1,2,4,6,7,9,12 2 pont

2: 3,5,8,10,11 2 pont

5. feladat: Szavak sorrendje (20 pont)

A. "TÍZ KILENC NYOLC HÉT HAT ÖT NÉGY HÁROM KETTŐ EGY" 5 pont

B. "EGY TÍZ KETTŐ KILENC HÁROM NYOLC NÉGY HÉT ÖT HAT" 5 pont

C. "EGY HÁROM ÖT HÉT KILENC" 5 pont

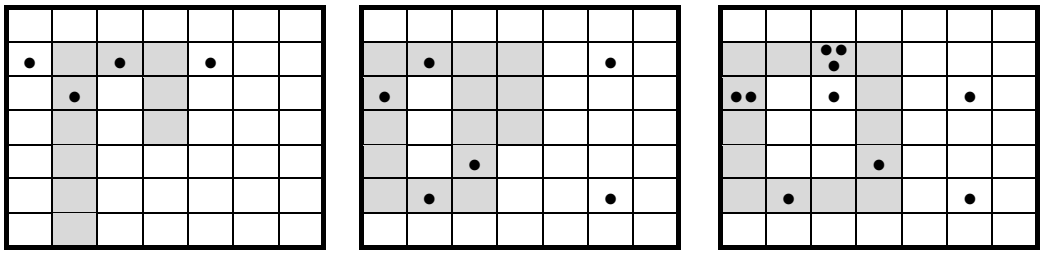
D. "EGY KETTŐ HÁROM NÉGY ÖT HAT HÉT NYOLC KILENC TÍZ KILENC NYOLC HÉT HAT ÖT NÉGY HÁROM KETTŐ EGY" 5 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Karesz a robot (18 pont)

A1. a 9. lépés után (a 10.-re már lelépne) B1. a 17. lépés után (a 18.-ra már lelépne) C1. a 19. lépés után (a 20.-ra már lelépne) 2+2+2 pont

A2-3. **B2-3.** **C2-3.**



Jó az útvonal 2+2+2 pont

A kövek jó helyen vannak 2+2+2 pont

2. feladat: Gráfalgoritmus (20 pont)

A. Minden csúcsra a belőle induló élek számát (minden hasonló megfogalmazás is jó, pl. a vele összekötött csúcsok száma) 4 pont

2,2,2,3,3,3,3,1,3,1,1 2 pont

B. Amelyekhez az adott lépésben már csak egyetlen él vezet 4 pont

8,10,11,7 2 pont

C. Amelyek körben vannak vagy köröket kötnek össze 3+3 pont

(1,2),(1,4),(2,4),(3,4),(3,5),(5,6),(5,9),(6,9) 2 pont

3. feladat: Szöveggyártás (15 pont)

Megjegyzés: az algoritmus a feltételnek megfelelő összes ismétléses permutációt állítja elő, ábécé sorrendben

A. Az angol ábécé kisbetűiből N betűt, amiben minden betű annyiszor szerepel, ami az `A` vektor megfelelő indexű elemében van, ábécé sorrendben 4+1 pont

B. Annyiszor, ahányféleképpen az `A` vektorral leírt betűk felsorolhatók 5 pont

(az ismétléses permutációk számára vonatkozó képlet is elfogadható: N!/(A('a')!*…*A('z')!))

C. Az egyes kiírások ábécé sorrend szerint növekvően követik egymást 5 pont

4. feladat: Prioritási sor (21 pont)

A. 5
4,5
3,4,5
3,4,5,6
4,5,6,8
5,5,6,8
5,6,8,9
6,7,8,9
7,8,9
8,9
9 lépésenként 1-1 pont=8 pont

B. Egyetlen elemet se előzzön meg a nála nagyobbak közül K vagy annál több 5 pont

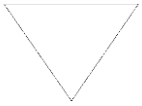
C1. (N-1)/(K-1) felső egész része (egy menetben legfeljebb K-1 helyet jöhet előre minden elem, s elképzelhető, hogy valamelyiknek N-1-et kellene előre lépni) 5 pont

Azonosan jó megoldás az (N+K-3)/(K-1) alsó egészrésze.

C2. A legrosszabb eset: ha a legkisebb elem kezdetben a sorozat végén volt 3 pont

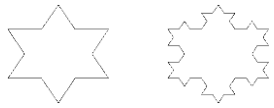
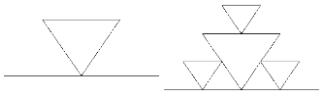
Azonosan jó megoldás, ha a legkisebb elem legalább az (((N+K-3)/(K-1))-1)*(K-1)+2. pozíción volt. ((A) az A szám alsó egészrészét jelöli.)

5. feladat: Fraktál (26 pont)

A1.  A2. 2+2 pont

B1. F+F--F+F--F+F--F+F--F+F--F+F--F+F 2 pont
F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F--F+F--F+F
+F+F--F+F--F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F 4 pont

B2. FXF+FXF-FXF-FXF+FXF 2 pont
FXF+FXF-FXF-FXF+FXF+FXF+FXF-FXF-FXF+FXF-FXF+FXF-FXF-FXF+FXF-
FXF+FXF-FXF-FXF+FXF+FXF+FXF-FXF-FXF+FXF 4 pont

C1.  C2.  2+3+2+3 pont

2000. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Tükördátumok (25 pont)

Egy adott évbeli tükördátumok az alábbi fajtájúak lehetnek:

a) XY.Z.YX, ha YX az hónapban legális napsorszám, Z pedig egyszámjegyű hónapsorszám;
b) XY.11.YX, ha YX az hónapban legális napsorszám;
c) XY.Y.X, ha Y legális egyjegyű hónapsorszám;
d) XY.1Y.X, ha Y=0 vagy 1 vagy 2;
e) X.1Y.1X, ha Y=0 vagy 1 vagy 2;
f) X.Y.YX, ha Y legális egyjegyű hónapsorszám és YX az hónapban legális napsorszám;
g) X.Y.X; ha Y legális egyjegyű hónapsorszám;

h) X.11.X.

A megoldást tehát aszerint bontjuk ketté, hogy az év egy- vagy kétszámjegyű. Az egyszámjegyű éveknél a következő algoritmust alkalmazzuk:

```
Egyszámjegyű(év):
  nap:=év
  Ciklus hó='1'-től '9'-ig
    Ki: év,'.',hó,'.',nap              {g eset}
    Ha hó≤'2' vagy év='1' és hó='3'
       akkor Ki: év,'.',hó,'.',hó,nap  {f eset}
  Ciklus vége
  Ki: év,'.10.1',nap                     {e eset}
  Ki: év,'.11.',nap                      {h eset}
  Ki: év,'.11.1',nap                     {e eset}
  Ki: év,'.12.1',nap                     {e eset}
Eljárás vége.
```

A kétszámjegyű éveknél biztosan csak a **c eset** fordulhat elő, ha az év második számjegye legalább 4. Ha az év 3-ra végződik, akkor **13.x.31** típusú lehet az eredmény a 31 napos hónapokra, továbbá **x3.3.x** mindegyikre. Az 1-re vagy 2-re végződő éveknél **x1.y.1x**, **x1.1.x**, illetve **x1.11.1x** típusú eredmény lehet.

```
Kétszámjegyű(év):
  Ha év(2)≥'4' akkor Ki: év,'.',év(2),'.',év(1)
  különben ha év(2)='3' akkor
             Ha év(1)='1' akkor
                Ciklus hó=1,3,5,7,8-ra
                  Ha hó=év(2) akkor Ki: év,'.',hó,'.',év(1)
                  Ki: év,'.',hó,'.',év(2),év(1)
                Ciklus vége
              különben Ki: év,'.',év(2),'.',év(1)
  különben ha év(2)∈{'1','2'} akkor
             Ciklus hó='1'-től '9'-ig
               Ha hó=év(2) akkor Ki: év,'.',év(2),'.',év(1)
               Ki: év,'.',hó,'.',év(2),év(1)
             Ciklus vége
             Ha év(2)='1' akkor Ki: év,'.1',év(2),'.',év(1)
                                Ki: év,'.11.',év(2),év(1)
             különben Ki: év,'.11.',év(2),év(1)
                      Ki: év,'.1',év(2),'.',év(1)
Eljárás vége.
```

2. feladat: Memóriajáték (24 pont)

Két részfeladatot kell megoldani. Az egyikben meg kell keresni az első eltérő karaktert a két karaktersorozatban.

```
Egyik(első,második,a,i):
  i:=1; a:=''
  Ciklus amíg i≤hossz(első) és i≤hossz(második)
             és első(i)=második(i)
    a:=a+első(i); i:=i+1
  Ciklus vége
Eljárás vége.
```

A másodikban a hasonlítás során az első karaktersorozatban akkor is tovább kell lépni, ha az nem egyezik meg a második megfelelő karakterével. Kihasználhatjuk azt, hogy az első részfeladatban eljutottunk az első eltérő karakterig.

```
Másik(első,második,b,i):
  j:=i; b:=''
  Ciklus amíg i≤hossz(első) és j≤hossz(második)
    Ha első(i)=második(j) akkor b:=b+első(i); j:=j+1
    i:=i+1
  Ciklus vége
Eljárás vége.
```

3. feladat: Programozási verseny (26 pont)

A bemenő adatokat az `n` elemű `x` tömbben kapjuk meg. A három részfeladatból az A és a B részfeladat egyszerre oldható meg. Meg kell vizsgálni, hogy hány különböző nyelv van és melyek ezek. Az A részfeladat megoldása a `db` változó, a B megoldása pedig az `ny` tömb lesz.

```
Nyelvek(db,ny,x,n):
  db:=1; ny(1).db:=1; ny(1).nyelv:=x(1).nyelv
  Ciklus i=2-től n-ig
    j:=1
    Ciklus amíg j≤db és x(i).nyelv≠ny(j).nyelv
      j:=j+1
    Ciklus vége
    Ha j>db akkor db:=db+1; ny(db).db:=1
                  ny(db).nyelv:=x(i).nyelv
          különben ny(j).db:=ny(j).db
  Ciklus vége
Eljárás vége.
```

A C részfeladat az eddigiek alapján `db` darab kiválogatás.

```
Versenyzők(db,ny,x,n):
  Ciklus i=1-től db-ig
    Ki: ny(i).nyelv
    Ciklus j=1-től n-ig
      Ha x(j).nyelv=ny(i).nyelv akkor Ki: x(j).név
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Gombaszedés (22 pont)

Első lépésként még a beolvasás közben érdemes a gombákat fajtánként 3 tömbbe válogatni, méghozzá nagyság szerint csökkenő sorrendben. A következő algoritmusnak e három tömb alapján adja meg az eredményt (A `Csz` elemű `C` tömbben lesz a csiperkegombák mérete, az `Rsz` elemű `R` tömbben a rókagombáké, az `Lsz` méretű `L` tömbben pedig a lila pereszkéké).

A feladat megoldása egy összefuttatás-szerű algoritmus lesz. Ha van még csiperke, akkor a feltételek szerint csak a csiperkéket és a rókagombákat kell vizsgálni. Ha már nincs csiperke, akkor jöhetnek a lila pereszkék és ha van még, akkor a rókagombák.

Mind a három tömb végére teszünk egy fiktív nullelemet, hogy az összefuttatás algoritmusa egyszerűbb lehessen.

```
  H:=0; Ci:=1; Ri:=1; Li:=1
  BeC:=0;BeR:=0;BeL:=0
  CS:=0; RS:=0; LS:=0
  C(Csz+1):=0; R(Rsz+1):=0; L(Lsz+1):=0
  Ciklus amíg Ci≤Csz          {amíg van csiperke}
    Ha C(Ci)≥ R(Ri) akkor
      Ha H+C(Ci)≤K
         akkor H:=H+C(Ci); BeC:=BeC+1; CS:=CS+C(Ci)
      Ci:=Ci+1
    különben ha H+R(Ri)≤K
         akkor H:=H+R(Ri); BeR:=BeR+1; RS:=RS+R(Ri)
      Ri:=Ri+1
  Ciklus vége
  Ciklus amíg Ri≤Rsz vagy Li≤Lsz {nincs már csiperke}
    Ha R(Ri)≥L(Li) akkor
      Ha H+R(Ri)≤K
         akkor H:=H+R(Ri); BeR:=BeR+1; RS:=RS+R(Ri)
      Ri:=Ri+1
    különben ha H+L(Li)≤K
         akkor H:=H+L(Li); BeL:BeL+1; LS:=LS+L(Li)
      Li:=Li+1
  Ciklus vége
Eljárás vége.
```

2. feladat: Hálózat (20 pont)

A feladatban szereplő gráf egy fa. Emiatt egy fában levő leghosszabb út hosszát kell meghatároznunk. Ezt két lépésben tehetjük meg. Először határozzuk meg az első ponttól legmesszebb levő pontot. Ez lesz a leghosszabb út egyik végpontja. Ezután pedig határozzuk meg az ettől a ponttól legmesszebb levő pontot. Ez lesz az út másik végpontja.

Könnyű belátni, hogy a fenti állítás igaz. Ha ugyanis nem az elsőtől legmesszebb levő pont lenne a leghosszabb út egyik végpontja, akkor az a végpont lecserélhető lenne az elsőtől legmesszebb levő pontra, és így az út hossza nőne, ami ellentmond annak a feltevésnek, hogy két másik pont a leghosszabb út végpontja.

Legyen `Apa(i)` az a gép, amelyhez i. gép kapcsolódik! Már a beolvasás közben kitölthetjük ezt a tömböt, valamint a feladat feltételei miatt (a következő gépet valamelyik korábbihoz kötik) számolhatjuk, hogy az egyes gépek milyen messze vannak az első géptől (`Táv(i)`). Jelöljük `M`-mel ezek közül azt a gépet, amelyik a legmesszebb van az elsőtől!

```
Beolvasás(Apa,M):
  Apa(1):=0; Táv(1):=0; M:=1
  Ciklus i=2-től N-ig
    Olvas(BeF,x); Apa(i):=x; Táv(i):=Táv(x)+1
    Ha Táv(i)>Táv(M) akkor M:=i
  Ciklus vége
Eljárás vége.
```

Könnyű kiszámolni az elsőtől legtávolabbi gép távolságát azoktól, amelyeken keresztül az elsőhöz van kötve. Kiindulunk az M-edikből, majd a távolságot egyesével növelve az Apa tömbön keresztül haladunk addig, amíg az elsőhöz ne érünk.

A többi pontnál viszont meg kell határozni, hogy az első menetben kapott pontok közül melyikhez van csatolva és milyen távolságra. Az M-ediktől mért távolsága a közös „ős" távolsága tőle, valamint az M-ediktől.

A közös ősig vezető úton levő távolságok megint könnyen kiszámolhatók, újra végighaladva a közös ősig vezető úton.

```
Leghosszabb út(Apa,M,MAxTáv):
  Táv:=0; x:=M; t:=0
  Ciklus amíg x>0 {az 1-hez vezető úton levők távolsága}
    Táv(x):=t; t:=t+1; x:=Apa(x)
  Ciklus vége
  MaxTáv:=Táv(1)
  Ciklus i=2-től N-ig
    Ha Táv(i)=0      {még nincs távolsága M-től}
      akkor x:=i; t:=0     {közös ős keresése}
            Ciklus amíg Táv(x)=0
              t:=t+1; x:=Apa(x)
            Ciklus vége
            UjTáv:=Táv(x)+t
            Ha UjTáv>MaxTáv akkor MaxTáv:=UjTáv
            x:=i  {a közös ősig levők távolsága}
            Ciklus amíg Táv(x)=0
              Táv(x):=Ujtáv; UjTáv:=UjTáv-1; x:=Apa(x)
            Ciklus vége
  Ciklus vége
Eljárás vége.
```

3. feladat: Zárnyitogató (12 pont)

Mind a három tárcsa forgatható balra is és jobbra is. Emiatt 8 esetet különböztethetünk meg. Minden tárcsa minden irányára számoljuk ki, hogy azt a tárcsát mennyivel kell elforgatni az (1,1,1) állapot eléréséhez (`L1`, `L2`, `L3`). Mivel a tárcsák együtt is forgathatók, ezért az egy irányba forgatás esetén ezen számok maximumára van szükség. Ha különböző irányba forgatjuk őket, akkor pedig az azonos irányúak maximumához kell hozzáadnunk az ellenkező irányba forgatott forgatásszámát.

```
Zár:
  Lépés:=2*N   {0= csökkenő, 1= növekvő irányba forog}
  Ciklus i1=0-tól 1-ig
    Ciklus i2=0-tól 1-ig
      Ciklus i3=0-tól 1-ig
        Ha i1=0 akkor L1:=A-1 különben L1:=N-A+1
        Ha i2=0 akkor L2:=B-1 különben L2:=N-B+1
        Ha i3=0 akkor L3:=C-1 különben L3:=N-C+1
        Ha i1=i2 és i2=i3 akkor L:=Max(L1,L2,L3)
        különben ha i1=i2 akkor L:=L3+Max(L1,L2)
        különben ha i1=i3 akkor L:=L2+Max(L1,L3)
        különben ha i2=i3 akkor L:=L1+Max(L2,L3)
        Ha L<Lépes akkor Lépes:=L
      Ciklus vége
    Ciklus vége
    Ciklus vége
Eljárás vége.
```

4. feladat: Lift (21 pont)

Jelölje `E(i,j)` azt, hogy az i-edik emeletről a j-edik utas melyik emeletre szeretne menni! Számoljuk ki először azt, hogy az a-adik emeletre hányan jönnének felfelé menő lifttel (`CF(a)`), illetve lefelé menő lifttel (`CL(a)`), valamint, hogy hányan mennének az i-edik emeletről felfelé (`F(i)`), illetve lefelé(`L(i)`)!

```
Számolás(CF,CL,F,L):
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg E(i,j)>0
      Ha E(i,j)>i akkor CF(E(i,j)):=CF(E(i,j))+1
                        F(i):=F(i)+1
                  különben CL(E(i,j)):=CL(E(i,j))+1
                        L(i):=L(i)+1
      j:=j+1
    Ciklus vége
Eljárás vége.
```

Ezután emeletenként számoljuk, hogy hányan szeretnének az adott emeletről felfelé menni, s az elszállításukhoz a liftnek hány menetre van szüksége!

Ehhez annyit kell tennünk, hogy az előző emeletről jövők számából levonjuk az itt kiszállók számát, majd hozzáadjuk az innen felfelé menni szándékozók számát.

```
Felfelémenők(CF,CL,F,L,Menet):
  Menet:=0; Fel:=0
  Ciklus i=1-től N-ig
    Fel:=Fel-CF(i)+F(i); H:=(Fel+K-1) div K
    Ha H>Menet akkor Menet:=H
  Ciklus vége
Eljárás vége.
```

A lefelé menőkkel hasonlóan járunk el:

```
Lefelémenők(CF,CL,F,L,Menet):
  Le:=0
  Ciklus i=N-től 1-ig -1-esével
    Le:=Le-CL(i)+L(i); H:=(Le+K-1) div K
    Ha H>Menet akkor Menet:=H
  Ciklus vége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Lift (15 pont)

A feladat megoldása ugyanaz, mint a kilencedik-tizedik osztályosok 4. feladatának megoldása. Az adott versenyévben annyi különbség volt a két feladat között, hogy az `E` mátrix a kisebbeknél elfért a számítógép memóriájában, a nagyobbaknál pedig nem. Azaz náluk a fenti vektorok (`CF`,`CL`,`F`,`L`) előállítását már az adatok beolvasásakor el kellett végezni, s emiatt nem volt szükség az `E` mátrixra.

2. feladat: Kockavilág (15 pont)

A feladat különlegessége, hogy a bemenet lehet tartalmilag hibás, ezért ezt is ellenőrizni kell. Kétféle tartalmi hiba fordulhat elő: ugyanarra a kockára többet is próbálunk rátenni, illetve az egymásra pakolás ciklikus, azaz egy kockát közvetve önmagára kellene tenni.

A beolvasáskor töltsük ki az `Alatt` tömböt, amely az i indexhez megadja, hogy az i-edik kocka alatt hányadik kocka van!

Az ellenőrzés az alábbi lehet:

```
Ellenőrzés:
  OK:=igaz
  Ciklus i=1-től N-ig
    x:=Alatt(i)
    Ha x>0 akkor OK:=OK és (Felett(x)=0); Felett(x):=i
  Ciklus vége
  Ha OK akkor
    szaml:=0
    Ciklus i=1-től N-ig
      Ha Felett(i)=0 akkor j:=i
                           Ciklus
                             szaml:=szaml+1; j:=Alatt(j)
                           amíg j>0
                           Ciklus vége
    Ciklus vége
    OK:=szaml=N
  Elágazás vége
Eljárás vége.
```

Első lépésként lehetséges, hogy a kezdőállapot tartalmazza részben a végeredményt (első legalul, rajta a második, rajta a harmadik, …), ezeket nem pakoljuk sehova.

```
Pakolás:
  i:=0
  Ciklus amíg i<N és Alatt(i+1)=i
    i:=i+1
  Ciklus vége
```

Második lépésként, ha ezen a jól induló oszlopon vannak még kockák, akkor azokat lerakjuk az asztalra.

```
  Ha i>0 akkor j:=i
               Ciklus amíg Felett(j)>0
                 j:=Felett(j)
               Ciklus vége
               Ciklus amíg j≠i
                 Rak(j,0); j:=Alatt(j); Felett(j):=0
               Ciklus vége
               Felett(i):=0
  Elágazás vége
```

Harmadik lépésként, ha az i-edik kockáig felépült a torony, akkor az i+1-edik kockáról le kell pakolni az asztalra, ami esetleg rajta van, majd pedig áttehető az i-edik kockára.

```
  Ciklus i=i+1-től N-ig
    j:=i
    Ciklus amíg Felett(j)>0
      j:=Felett(j)
    Ciklus vége
    Ciklus amíg j≠i
      Rak(j,0); j:=Alatt(j); Felett(j):=0
    Ciklus vége
    Rak(i,i-1); j:=Alatt(i)
    Ha j>0 akkor Felett(j):=0
  Ciklus vége
Eljárás vége.
```

A megoldásban a `Rak(mit,mire)` eljárás írja ki az eredménybe, hogy melyik kockát melyikre kellett rakni.

3. feladat: Malacpersely(15 pont)

A feladat szerint adott összeghez (`Suly`) meg kell határozni hogy adott számok közül (pénzérmék súlya) melyek összege lehet, ráadásul úgy, hogy ezek összértéke minimális legyen.

Írjunk egy függvényt, amely megadja az `X` súlyhoz tartozó legkisebb pénz összeget, jelöljük ezt `E(X)`-szel: $E(X)=\min_{i=1,..N}(E(X-S_i)+P_i)$. Ez a képlet rekurzívan is kiszámítható, de figyelni kell arra, hogy ugyanazt az értéket ne számoljuk ki kétszer, azaz a közbülső kiszámított értékeket tárolni kell, s egy adott értéket csak akkor számolunk ki, ha korábban még nem számoltuk.

```
Mini(X):
  Ha E(X)≥0 akkor Mini:=E(X)
  különben M:=MaxP
           Ciklus i=1-től N-ig
             Ha X≥S(i) akkor Mx:=Mini(X-S(i))
                             Ha Mx+P(i)<M akkor M:=Mx+P(i)
           Ciklus vége
           E(X):=M; Mini:=M
  Elágazás vége
Eljárás vége.
```

Ezután a főprogram szinte egyetlen eljáráshívásból áll. Előtte az `E` tömböt be kell állítani úgy, hogy még minden értéket ki kelljen számolni, továbbá akkor nincs megoldás (-1 értéket kell kiírni), ha nem állítható elő az adott érmékkel az adott súly.

```
Számítás:
  Ciklus i=1-től Súly-ig
    E(i):=-1
  Ciklus vége
  E(0):=0; Megold:=Mini(Suly)
  Ha Megold≥MaxP akkor Megold:=-1
Eljárás vége.
```

4. feladat: Raktár (15 pont)

A beolvasás során a sorfolytonosan utolsó pont helyét az (`x0`,`y0`) változókban tároljuk, a pont karaktereket `H(i,j)=0`-val, a #-karaktereket pedig `H(i,j)=1`-gyel kódoljuk.

A leghosszabb útvonal egyik végpontja biztosan az (`x0`,`y0`) ponttól legtávolabbi pont lesz. Ha ezt megtaláltuk, akkor a leghosszabb út másik végpontja az ettől a ponttól legtávolabbi pont lesz.

```
Leghosszabb út(MaxTav):
  Legmesszebb(x0,y0,xt,yt,MaxTav)
  Legmesszebb(xt,yt,xx,yy,MaxTav)
Eljárás vége.
```

Egy adott ponttól legtávolabbi pont megtalálásához pedig egy szélességi gráfbejárást alkalmazunk. A gráfbejárás során elért pontokat 0-ról átkódoljuk 2-re (majd a bejárás végén visszaalakítjuk 0-ra a következő bejárás miatt). A szomszédok kiszámításához az `Sz` indextömböt használjuk.

```
Sz: Tömb(1..4, rekord x,y:egész)
            =((x:1;y:0),(x:0;y:1),(x:-1;y:0),(x:0;y:-1))
```

```
Legmesszebb(x0,y0,xt,yt,MaxTav):
  SorInicializálás; Sorba(x0,y0,0); H(x0,y0):=2; MaxTav:=0
  Ciklus amíg a Sor nem üres
    Sorból(x,y,t)
    Ha t>MaxTav akkor MaxTav:=t; xt:=x; yt:=y
    Ciklus i=1-től 4-ig       {a négy lehetséges szomszéd}
      xx:=x+Sz(i).x; yy:=y+Sz(i).y
      Ha H(xx,yy)=0 akkor H(xx,yy):=2; Sorba(xx,yy,t+1)
    Ciklus vége
  Ciklus vége
  Ciklus x=1-től N-ig
    Ciklus y=1-től M-ig
      Ha H(x,y)=2 akkor H(x,y):=0
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

5. feladat: Üzenetek (15 pont)

A feladat megoldása tulajdonképpen egy szabályosan zárójelezett kifejezés elemzését jelenti, a kifejezés fa struktúrában is elképzelhető.

Az A részfeladat az egy szinten (azonos zárójelben) levő elemek maximális számát kéri (`MaxSz`). A B részfeladat megoldása a zárójelezés maximális mélységének kiszámítása (`MaxM`). A C részfeladat pedig azon ún. levélelemek száma, amelyet nem követ nyitózárójel (`Levél`).

Jelölje `M` az aktuális zárójelezési mélységet, az `SZ(M)` pedig az aktuális mélységhez tartozó elemek számát!

```
Számolás:
  MaxM:=0; MaxSz:=0; M:=0; Levél:=0
  Ciklus
    Olvas(BeF,X)
    Ha X ∉('(',')',',','#')
       akkor Levél:=Levél+1
             Ciklus
               Olvas(BeF,X)
             amíg X ∉('(',')',',','#')
             Ciklus vége
    Elágazás vége
    Elágazás
      X='(' esetén M:=M+1; Levél:=Levél-1; Sz(M):=1
                   Ha M>MaxM akkor MaxM:=M
      X=')' esetén M:=M-1
                   Ha Sz(Vm)>MaxSz akkor MaxSz:=Sz(Vm)
      X=',' esetén Sz(M):=Sz(M)+1
    Elágazás vége
  amíg X≠'#'
  Ciklus vége
Eljárás vége.
```

2000. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Villamos (25 pont)

Jelölje `fel(i)` az i-edik állomáson felszállók, `le(i)` pedig a leszállók számát! Az egyik részfeladat a tartalmi helyesség ellenőrzése.

Tartalmilag hibásak az adatok, ha az induló állomáson volt leszálló, illetve ha a végállomáson volt felszálló. Ugyancsak hibás, ha egy állomáson a leszállók száma nagyobb, mint az aktuális utasszám. Akkor is hibásak az adatok, ha a végállomáson a leszállás után maradnak a villamoson utasok.

Ha az adatok helyesek, akkor a villamoson utazók össz száma azonos a felszállók össz számával (`osszes`), vagy a leszállókéval, hiszen a két összegnek azonosnak kell lenni. Másodikként ki kell válogatni azon állomások sorszámát, amikor a leszállás után (a felszállás előtt) az utasszám 0 lett (`db`,`hely`). Ehhez természetesen számolni kell a pillanatnyi utasszámot (`utasszam`). Ez utóbbi maximuma lesz a C részfeladat megoldása (`max`). A D részfeladat megoldásához azon esetek számát kell megadni, amikor a felszállások után 0 volt az aktuális utasszám.

```
Villamos:
  jo:=igaz
  Ha le(1)>0 vagy fel(n)>0 akkor jo:=hamis
  utasszam:=0; osszes:=0; max:=0; db:=0; uresek:=0
  Ciklus i=1-től n-ig
    Ha utasszam<le(i) akkor jo:=hamis
    utasszam:=utasszam-le(i)
    Ha utasszam=0 és i>1 és le(i)≠0
                    akkor db:=db+1; hely(db):=i
    osszes:=osszes+fel(i); utasszam:=utasszam+fel(i)
    Ha utasszam>max akkor max:=utasszam
    Ha utasszam=0 és i<n akkor uresek:=uresek+1
  Ciklus vége
  Ha utasszam>0 akkor jo:=hamis
Eljárás vége.
```

2. feladat: Mondat-csavaró (20 pont)

A beolvasott mondatot tegyük a `mondat` változóba! Az első részfeladat megoldásához olvassuk hátulról a mondat betűit, és gyűjtsük egyesével egy szóba (`szo`)! Ha szóközhöz érünk, akkor az eddig elkészült szó kiírható, majd új szót kezdhetünk.

```
A részfeladat:
  szo:=''
  Ciklus i=hossz(mondat)-tól 1-ig -1-esével
    Ha mondat(i)=' ' akkor Ki:szo,' '; szo:=''
                   különben szo:=mondat(i)+szo
  Ciklus vége
  Ki: szo
Eljárás vége.
```

A második részfeladatnál a páratlan sorszámú szavakat kell kiírni, ezért csak arra van szükségünk, hogy az adott betű páratlan sorszámú szóhoz tartozik-e (`paratlan`), s ha igen, akkor a betű kiírható.

```
B részfeladat:
  paratlan:=igaz
  Ciklus i=1-től hossz(mondat)-ig
    Ha paratlan akkor Ki: mondat(i)
    Ha mondat(i)=' ' akkor paratlan:=nem paratlan
  Ciklus vége
Eljárás vége.
```

A harmadik részfeladatnál azt kell figyelnünk, hogy szó kezdőbetűjénél vagyunk-e (`elso`), majd azokat nagybetűsen kiírni. Kezdőbetű a mondat első betűje, majd pedig minden szóköz utáni betű.

```
C részfeladat:
  elso:=igaz
  Ciklus i=1-től hossz(mondat)-ig
    Ha elso akkor Ki: nagybetűs(mondat(i)),' '
    elso:=(mondat(i)=' ')
  Ciklus vége
Eljárás vége.
```

A negyedik részeladatnál a szavakat gyűjtjük ki (`szo`), de a betűk sorrendjét megfordítjuk, majd a szó végére érve kiírjuk.

```
D részfeladat:
  szo:=''
  Ciklus i=1-től hossz(mondat)-ig
    Ha mondat(i)≠' ' akkor szo:=mondat(i)+szo
                   különben Ki: szo,' '; szo:=''
  Ciklus vége
  Ki: szo
Eljárás vége.
```

3. feladat: Húsvét (30 pont)

A feladat megoldásához szükségünk van az aktuális évre (`Ev`). Tudnunk kell, hogy az év első napja a hét hányadik napjára esik (`Elso`). Szükségünk van arra, hogy az első holdtölte az év hányadik napjára esik (`Hold`), valamint, hogy ez délelőtt volt-e (`De`).

A feladat szerint ugyanis: Húsvét a tavaszi napéjegyenlőség (március 21.) utáni első holdtölte utáni első vasárnap, illetve hétfő. A holdtölték egymástól 29 és fél napra vannak. Pünkösdvasárnap a húsvétvasárnap utáni hetedik vasárnap, pünkösdhétfő pedig az azt követő hétfő.

Először meg kell határoznunk, hogy a tavaszi napéjegyenlőség az év hányadik napjára esik (`Napej`), figyelve a szökőévre is, továbbá szükségünk van az első vasárnap sorszámára is (`Elso`). Ezután ki kell számolni, hogy hányadik napon van az ezutáni első holdtölte (`Hold`). Ezután megkeressük a következő vasárnapot.

Most már ismerjük a húsvétvasárnap sorszámát az éven belül, ebből kell meghatározni a hónap nevét (`Husvet`) és a hónapon belüli nap sorszámot (`Nap`). A húsvéthétfő napsorszáma eggyel nagyobb, illetve ha a vasárnap március 31 volt, akkor a hétfő április 1 lesz.

A pünkösd éven belüli napsorszáma pontosan 49-cel nagyobb a húsvéténál, azt kell csak eldönteni, hogy májusban vagy júniusban van-e (`Punkosd`). A pünkösdhétfőt ugyanúgy kell meghatározni, mint a húsvéthétfőt.

```
Húsvét:
  Elso:=(8-Elso) mod 7;     {Az első vasárnap}
  Napej:=31+28+21;          {A tavaszi napejegyenloseg}
  Ha Ev mod 4=0 és Ev mod 100≠0 vagy Ev mod 400=0
     akkor Napej:=Napej+1
  Ciklus amíg Hold<Napej       {Az első holdtölte utána}
    Hold:=Hold+29; Ha nem De akkor Hold:=Hold+1
    De:= nem De
  Ciklus vége
  Ciklus amíg Hold mod 7≠Elso    {A következő vasárnap}
    Hold:=Hold+1
  Ciklus vége
  Ha Hold<Napej+11
       akkor Husvet:='március '; Nap:=Hold-Napej+21
    különben Husvet:='április '; Nap:=Hold-Napej-10
  Ki: 'Húsvétvasárnap: ',Husvet,Nap
  Ha Nap<31 akkor Ki: 'Húsvéthétfő: ',Husvet,Nap+1
          különben Ki: 'Húsvéthétfő: április 1.'
  Hold:=Hold+49
  Ha Hold<Napej+72
        akkor Punkosd:='május '; Nap:=Hold-Napej-40
     különben Punkosd:='június '; Nap:=Hold-Napej-71
  Ki: 'Pünkösdvasárnap: ',Punkosd,Nap
  Ha Nap<31 akkor Ki: 'Pünkösdhétfő: ',Punkosd,Nap+1
          különben Ki: 'Pünkösdhétfő: június 1.'
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Autópálya (23 pont)

Jelöljük `U(i)`-vel azt a távolságot, amit az i. kútnál levő benzinnel megtehetünk, `T(i)`-vel pedig az i. és i+1. (cirkulárisan) kút távolságát. Először keressük meg a legkisebb k-t, amelyik kúttól körbejárhatunk, azaz $\sum_{i=k\to k-1} U_i \geq \sum_{i=k\to k-1} T_i$

```
   U(k)      U(k+1)      . . .                          U(k-1)
  0------->0------->0----->0------->0----->0------>0----->0
    T(k)     T(k+1)                                   T(k-1)
  0----->0------->0----->0------->0----->0----->0------->0
```

Ha nincs ilyen, akkor jegyezzük fel a leghosszabb olyan részsorozat hosszát és kezdetét, amelyre még teljesült az egyenlőtlenség. Ez a k úgy számolható, hogy képezzük a fenti összegeket addig, amíg az egyenlőtlenség fennáll. Ha i cirkulárisan eléri k-t, akkor az a k jó, egyébként folytassuk a keresést k-t i+1-nek véve. Az biztos, hogy k-nál kisebb kúttól nem lehet körbeautózni. A többi helyet úgy határozhatjuk meg, hogy k-tól (cirkulárisan) visszafelé megkeressük azt a legnagyobb i-t amelytől el lehet jutni k-ba. Ekkor i-től is körbejárhatunk. i-ből akkor és csak akkor juthatunk el k-ba, ha U(k-1)+...+U(i)≥ T(k-1)+...+T(i). Majd ezután azt a helyet keressük, amelytől ezen i-ig el tudunk jutni, ez is megoldás lesz, és így tovább, míg vissza nem érünk k-hoz.

```
Körút:
  i:=1; Maxh:=0
  Ciklus          {k.-tól indulva körbe lehet-e menni?}
    k:=i; oT:=0; oU:=0; h:=0
    Ciklus
      h:=h+1; oT:=oT+T(i); oU:=oU+U(i)
      i:=i+1; Ha i>N akkor i:=1
    amíg oT≤oU és k≠i
    Ciklus vége
    Ha h-1>Maxh akkor Maxh:=h-1; k0:=k
  amíg i>k
  Ciklus vége
  Ha oT>oU akkor Ír(KiF,'NEM'); Ír(KiF, k0)
  különben {k-tól indulva körbejárható}
           Ír(KiF,'IGEN'); Ír(KiF, k:1)
           i:=k
           Ciklus
             oT:=0; oU:=0
             Ciklus
               i:=i-1; Ha i=0 akkor i:=N
               oT:=oT+T(i); oU:=oU+U(i)
             amíg i≠k és oT>oU
             Ciklus vége
             Ha k≠i akkor Ír(KiF,' ',i:1)
           amíg k≠i
  Elágazás vége
Eljárás vége.
```

A feladat másképpen is megoldható. A körbejárás feltétele lehet az, hogy van-e összesen annyi benzin, ami elég a körbejáráshoz. Ezután elég minden helyre azt tárolni, hogy a maradék benzin (ami nem a fogy el a következő benzinkútig) mekkora – negatív is lehet, ha az adott benzinkútnál kevesebb benzin volt, mint ami a következőig elég (`maradek`).

Ha körbe lehet járni, akkor meg kell keresni az első helyet, ahonnan a `maradek` vektor elemeit ciklikusan összeadva a részösszegek folyamatosan nemnegatívok.

```
Elsőhely(k:):
  k:=1; körbeért:=hamis
  Ciklus amíg nem körbeért
    i:=k; s:=maradek(k)
    Ciklus amíg s≥0 és nem körbeért
      i:=i+1; Ha i>n akkor i:=1
      Ha i=k akkor körbeért:=igaz különben s:=s+maradek(i)
    Ciklus vége
    Ha s<0 akkor k:=i+1; ha k>n akkor k:=1;
  Ciklus vége
Eljárás vége.
```

Ezután a többi ilyen hely megkeresése a következőképpen történhet: Ha az előző hely maradéka nemnegatív, akkor abból indulva is biztosan körbe lehet járni. Ha az előző hely maradéka negatív, akkor haladjunk visszafelé addig, összegezve a maradékokat, amíg az összeg nem lesz újra pozitív. Az ilyen pontból újra körbe lehet járni. Mindezt ismételjük addig, amíg vissza nem érünk az első jó pontba.

```
Előzők(k):
  i:=k-1; Ha i=0 akkor i:=n
  Ciklus amíg i≠k
    Ha maradek(i)≥0 akkor Ír(g,i,' ')
                          i:=i-1; Ha i=0 akkor i:=n
    különben {ide el lehet-e érni valahonnan}
             s:=maradek(i)
             Ciklus amíg s<0 és i≠k
               i:=i-1; Ha i=0 akkor i:=n
               Ha i≠k akkor s:=s+maradek(i)
             Ciklus vége
             Ha i≠k akkor {i-ből ér ide}
                          Ír(g,i,' ')
                          i:=i-1; Ha i=0 akkor i:=n
             Elágazás vége
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

Ha nem lehet körbejárni, akkor keressük meg azt a pontot, ahonnan a legtovább lehet menni, azaz a részletösszegek a legtovább nemnegatívak.

```
Leghosszabb(maxk):
  k:=1; maxk:=0; h:=0; vege:=hamis
  Ciklus
    i:=k; s:=maradek(k); db:=1
    Ciklus amíg s≥0
      i:=i+1; Ha i>n akkor i:=1; vege:=igaz
      s:=s+maradek(i); db:=db+1
    Ciklus vége
    Ha db>h akkor h:=db; maxk:=k
    k:=i+1
  amíg nem vege és k≤n
  Ciklus vége
Eljárás vége.
```

2. feladat: Vállalatok (30 pont)

A bemeneti állományt soronként beolvasva építsük fel azt az adatszerkezetet, amelyben egy Ősszam(V) tárolja, hogy az eddigi napig alakult vállalatokat tekintve a V vállalatnak hány őse volt. Az Ős(V) érték ekkor legyen egy olyan vállalat, amely olyan közvetlen őse volt V-nek, amelynek a legtöbb őse volt. Azt is feljegyezhetjük, hogy eddig a napig melyik vállalatnak volt a legtöbb őse. Ugyancsak számoljuk azt, hogy aktuálisan hány létező vállalat van, illetve ezek maximumát, és hogy melyik napon volt a maximum.

```
Vállalatok:
  Olvas(Bef,M,c,Nev(1))
  Letezo:=1; Osszam(1):=1; Os(1):=0; N:=1
  MaxLetezo:=1;MaxNap:=0; MaxO:=1; MaxV:=1
  Ciklus i=1-től M-ig
    Olvas(BeF,Valt,V,Nap)
    iV:=Keres(V)
    Ha iV=0 akkor          {új név}
      N:=N+1; Nev(N):=V; iV:=N
      Osszam(iV):=0; Os(iV):=0
    Elágazás vége
    MaxO1:=Osszam(iV); MaxV1:=iV
```

```
    Ha Valt='+' akkor
      Letezo:=Letezo+1
      Ciklus amíg nem sorvég(BeF)
        Olvas(W); iW:=Keres(W); Letezo:=Letezo-1
        Ha V≠W akkor Ha Osszam(iW)+1>MaxO1 akkor
                       MaxO1:=Osszam(iW)+1; Os(iV):=iW
                       Osszam(iV):=MaxO1; MaxV1:=iV
        Elágazások vége
      Ciklus vége
    különben {Valt='-'}
      Letezo:=Letezo-1
      Ciklus amíg nem sorvég(BeF)
        Olvas(W); iW:=Keres(W); Letezo:=Letezo+1
        Ha V≠W akkor N:=N+1; Nev(N):=W; iW:=N
                     Os(iW):=iV; Osszam(iW):=Osszam(iV)+1
                     MaxO1:=Osszam(iW); MaxV1:=iW
      Ciklus vége
    Elágazás vége
    Ha Letezo>MaxLetezo akkor MaxLetezo:=Letezo; MaxNap:=Nap
    Ha MaxO1>MaxO akkor MaxO:=MaxO1; MaxV:=MaxV1
  Ciklus vége
Eljárás vége.
```

3. feladat: Alkimisták (22 pont)

Tekintsük azt a gráfot, amelynek csúcsai az anyagok, és két anyag, p és q között akkor és csak akkor van p→q C-vel címkézett él, ha p-ből a C katalizátorral kapható a q anyag. Ez a gráf körmentes. Számítsuk ki minden p anyagra azon címkék (katalizátorok) halmazát, amelyek bármely 1-ből p-be vezető úton előfordulnak címkeként. Jelöljük ezt `katalizator(p)`-vel. A megoldás: a `szukseges(0)` halmaz elemei.

Lépésről-lépésre számítjuk a `katalizator(q)` halmazokat olyan sorrendben, hogy bármely olyan p-re, amelyre van p→q él, akkor p-re előbb számítsuk ki `katalizator(p)` értékét. Ilyen sorrend megállapítható, hiszen a gráf körmentes, így kezdhetünk az 1-ponttal, amire `szukseges(1)` üres halmaz.

Színezzük be szürkével azokat a p pontokat, amelyekre már kiszámítottuk `katalizator(p)` értékét, de lehet olyan p→q el, hogy q-ra meg nem számítottuk ki. Egy pont színe szürkéről feketére változik, ha minden szomszédja (ha van) szürke lett. Kezdetben minden pont színe legyen fehér, azaz érintetlen.

Minden lépésben válasszunk egy olyan p pontot, amely szürke, de nincs olyan u pont, hogy van u→p él és u színe nem fekete. p minden q szomszédjára számítsuk a `katalizator(q)` halmazt:

Ha q színe fehér, akkor katalizator(q):=katalizator(p)∪ a p→q él címkéje, ha q színe nem fehér, akkor katalizator(q):=katalizator(p)∩ (katalizator(p)∪+ p→q címkéje).

```
Útkeresés(x,y):
  Ciklus i=0-tól maxanyag-ig
    szin(i):=fehér; katalizator(i):=()
  Ciklus vége
  katalizator(x):={}; szin(x):=szürke
  Ciklus {szürke színű pont keresése 0 belépő éllel}
    i:=0
    Ciklus amíg i≤maxanyag és nem(befok(i)=0 és szin(i)=szürke)
      i:=i+1
    Ciklus vége
    Ha i≤maxanyag akkor
      Ciklus j=0-tól maxanyag-ig
        Ha cs(i,j)≠' ' akkor
          Ha szin(j)=fehér akkor
             szin(j):=szürke; befok(j):=befok(j)-1
             katalizator(j):=katalizator(i)∪{cs(i,j)}
          különben ha szin(j)=szürke akkor
             befok(j):=befok(j)-1
             katalizator(j):=katalizator(j)∩
                             (katalizator(i)∪{cs(i,j)})
      Ciklus vége
      szin(i):=fekete
    Elágazás vége
  amíg i≤maxanyag
  Ciklus vége
  Ha szin(y)≠fekete akkor Ír(g,'NEM LEHET')
  különben ha katalizator(y)={} akkor Ír(g,'EGYIK SEM KELL')
  különben Ciklus c='A'-tól 'Z'-ig
             Ha c∈katalizator(y) akkor Ír(g,c,' ')
           Ciklus vége
Eljárás vége.
```

Másik megoldás: Nincs semmilyen kikötés a bemenetre; a gráf nem feltétlenül körmentes és két anyagra több bejegyzés is lehet.

Tekintsük azt a gráfot, amelynek csúcsai az anyagok, és két anyag, `p` és `q` között akkor és csak akkor van `p`→`q` C-vel címkézett él, ha `p`-ből a C katalizátorral kapható a `q` anyag. Számítsuk ki minden `p` anyagra azon címkék (katalizátorok) halmazát, amelyek bármely `p`-ből 0-ba vezető úton előfordulnak címkeként. Jelöljük ezt `Kell(p)`-vel!

A megoldás: a `Kell(1)` halmaz elemei. A számítást a `Bejár` eljárás végzi rekurzívan. `Kell(0)` értéke legyen egy speciális jel (`Jel2`), hogy ne legyen üres. `Kell(p)`-t úgy számíthatjuk, hogy minden `q` szomszédjára, ha ez fehér, tehát még nem számítottuk ki a `Kell(q)` értéket, akkor előbb meghívjuk a `Bejár` eljárást `q`-ra (ami `Kell(q)` értékét számolja ki). Ha a szomszéd nem fehér, akkor már kiszámoltuk a `Kell(q)` értéket, amit a `Kell(q)` tömbben tárolunk.

```
Bejár(P):
  Ha P=0 akkor Kell(0):={Jel1}; Szin(0):=Fekete
  különben Szin(P):=Szürke
           Ciklus Q=0-tól MaxA-ig
             Ha Be(P,Q)≠Jel2 akkor
               Ha Szin(Q)=Fehér akkor Bejár(Q)
             Ha Kell(Q)≠{} akkor
               Ha Kell(P)={} akkor {Kell(P) még üres}
                  Kell(P):={Be(P,Q)}∪{Jel1}∪Kell(Q)
               különben
                  Kell(P):=Kell(P)∩({Be(P,Q)}∪Kell(Q))
             Elágazás vége
           Ciklus vége
           Szin(Q):=Fekete
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Térkép (30 pont)

A feladatban feltett három távolságra vonatkozó kérdésre ugyanazon algoritmussal számítható a megoldás, ha a mezők közötti távolságot, azaz egy lépés idejét az alábbiak szerint értelmezzük.

1.

	1	2
1	0	1
2	0	0

2.

	1	2
1	1	1
2	0	0

3.

	1	2
1	1	1
2	2	2

A két pont, (A,B) és (U,V) közötti legrövidebb út hosszát kell számítani a megfelelő távolság-értelmezés szerint.

A szomszédok relatív koordinátái:

```
Sz: Tömb(1..4,Rekord dx,dy:egész)=
    ((dx:1;dy:0),(dx:0;dy:1),(dx:-1;dy:0),(dx:0;dy:-1))
```

Egy lépés ideje a három távolság értelmezés szerint:

```
L: Tömb(1..3,'0'..'2','0'..'2',Word =(
                                    ((Inf,Inf,Inf),
                                     (Inf,  0,  1),
                                     (Inf,  0,  0)),
                                    ((Inf,Inf,Inf),
                                     (Inf,  1,  1),
                                     (Inf,  0,  0)),
                                    ((Inf,Inf,Inf),
                                     (Inf,  1,  1),
                                     (Inf,  2,  2)))
```

A gráf bejárásához egy prioritási sort használunk, a sorban a pontok a kezdőponttól vett távolságuk szerint helyezkednek el, s a bejárás során a sorban levő pontok távolsága, és emiatt a helye is változhat.

```
Megold(f,X0,Y0,X1,Y1):
  MinTav(,):=Inf; Hol(,):=0
  MinTav(x0,y0):=0; SorInicializálás
  x:=x0;y:=y0; Sorba(x0,y0)
  Ciklus amíg a sor nem üres és (x≠X1 vagy y≠Y1)
    Sorból(x,y)
    Ciklus ir=1-től 4-ig      {a négy szomszéd irányába}
      xx:=x+Sz(ir).dx; yy:=y+Sz(ir).dy
      UjTav:=MinTav(x,y)+L(f, G(x,y),G(xx,yy))
      Ha UjTav<MinTav(xx,yy) akkor MinTav(xx,yy):=UjTav
                                   Sormod(xx,yy)
    Ciklus vége
  Ciklus vége
  Ha MinTav(X1,Y1)≥Inf akkor {nincs (X0,Y0)→(X1,Y1) út!}
                             Megold:=-1
                   különben Megold:=MinTav(X1,Y1)
Eljárás vége.
```

Külön feladat a két adott pontot tartalmazó város területének meghatározása, ez tulajdonképpen egy gráfbejárás meghívását jelenti mindkét pontra.

```
Bejár(x,y,T):
  T:=T+1; G(x,y):='0'
  Ciklus ir=1-től 4-ig     {a négy szomszéd irányába}
    xx:=x+Sz(ir).dx; yy:=y+Sz(ir).dy
    Ha G(xx,yy)='2' akkor Bejár(xx,yy)
  Ciklus vége
Eljárás vége.
```

2. feladat: Karaván (30 pont)

Jelölje `Ind(X,i)` azt a legkésőbbi indulási időt, amikor el kellett indulni a kiindulási A városból, hogy legkésőbb az i-edik napra X-be érkezzünk! Ha nem lehet eljutni X-be az i. napig, akkor legyen `Ind(X,i)` értéke 0. Jelölje továbbá `Erk(X)` azt a napot, amikor legkorábban az X városba érkezhetünk.

`Ind(X,0)` értéke a kiindulási A városra 1, más X-re 0. Látható, hogy `Ind(X,i)` értéke kiszámítható az `Ind(Y,i-1)` értékek és a bemeneti adatok alapján, így elegendő a kiszámításhoz két egymást követő i-re tárolni az `Ind(X,i)` adatokat. `Erk(X)` párhuzamosan számítható `Ind(X,i)`-vel.

A második kérdésre a választ az `i-Ind(B,i)+1` értékek minimuma adja. A három részfeladat megoldását az `M1`, `M2`, `M3` változókba tesszük.

```
Karaván:
  M1:=Inf; M2:=Inf; M3:=0; Ma:=igaz; Tegnap:=hamis
  Ciklus x=1-től N-ig
    Ind(x,Tegnap):=0; Erk(x):=Inf
  Ciklus vége
  Ind(A,Tegnap):=0; Erk(A):=0
```

```
  Ciklus i=1-től MaxInd-ig
    Ind(A,Tegnap):=i
    Ciklus y=1-től N-ig
      Ha y≠A akkor
        yi:=Ind(y,Tegnap)
        Ciklus z=1-től BeFok(y)-ig
          x:=G(y,z).u; p:=G(y,z).p; q:=G(y,z).q
          Ha p≤i és i≤q és yi<Ind(x,Tegnap)
             akkor yi:=Ind(x,Tegnap);
        Ciklus vége
        Ind(y,Ma):=yi
        Ha yi>0 és i<Erk(y) akkor Erk(y):=i
    Ciklus vége
    Ha Ind(B,Ma)>0 és i-Ind(B,Ma)+1<M2
       akkor M2:=i-Ind(B,Ma)+1
    Ha i≤H és M3<Ind(B,Ma) akkor M3:=Ind(B,Ma)
    Tegnap:=Ma; Ma:=Nem Tegnap
  Ciklus vége
  M1:=Erk(B)
  Ha M1≥Inf akkor M1:=-1
  Ha M2≥Inf akkor M2:=-1
  Ha M3=0   akkor M3:=-1
Eljárás vége.
```

Másik megoldás. Jelölje `Érkezés(i)` azt a legkorábbi napot, amikorra a B célállomásra érkezhetünk, ha az A indulási helyről az i. napon elindulunk.

Minden lehetséges i-re kiszámolva `Érkezés(i)`-t, mindhárom kérdésre választ tudunk adni. Egy adott i-re `Érkezés(i)`-t úgy számíthatjuk, hogy minden X városra számítjuk, hogy oda legkorábban mikor érkezhetünk, feltéve, hogy az A városból az i. napon indultunk.

```
Karaván:
  M1:=Inf; M2:=Inf; M3:=Inf
  Ciklus i=1-től MaxInd-ig
    Nap:=Érkezés(i)
    Ha Nap<Inf akkor Ha Nap<M1 akkor M1:=Nap
                     Ha Nap-i+1<M2 akkor M2:=Nap-i+1
                     Ha Nap≤H akkor M3:=i
  Ciklus vége
  Ha M1≥Inf akkor M1:=-1
  Ha M2≥Inf akkor M2:=-1
  Ha M3≥Inf akkor M3:=-1
Eljárás vége.
```

Itt egy szabályos szélességi gráfbejárást alkalmazunk:

```
Érkezés(i):
  SorInicializás; Erk():=Inf; Erk(A):=iNap-1; Sorba(A)
  Ciklus amíg a Sor nem üres
    Sorból(X)
    Ciklus j=1-től KiFok(X)-ig
      Y:=G(X,j).u; Ex:=Erk(X)
      Ha Ex<G(X,j).q akkor
        Ey:=Ex+1; Ha Ey<G(X,j).p akkor Ey:=G(X,j).p
        Ha Ey<Erk(Y) akkor Erk(Y):=Ey; Sorba(Y)
    Ciklus vége
  Ciklus vége
  Érkezés:=Erk(B)
Eljárás vége.
```

3. feladat: Dominó (15 pont)

Jelöljük `T(u,v,i)`-vel annak a leghosszabb illeszkedő dominósornak a hosszát, amely kirakható az első i dominóból és egyik végén `u`, a másikon `v` pötty van. Nyilvánvaló, hogy a `T(u,v,i+1)` értékek kiszámíthatók az i+1-edik dominó [x,y] és `T(u,v,i)` értékekből, hiszen minden `u`-ra (0≤u≤9):

```
T(u,y,i+1)=Max{T(u,y,i),T(u,x,i)+1},
T(u,x,i+1)=Max{T(u,x,i),T(u,y,i)+1}.
```

```
Dominó:
  Olvas(BeF,X,Y)
  T[X,Y]:=1; T[Y,X]:=1; Tu:=T   {kirakjuk az első dominót}
  Ciklus i:=2-től N-ig
    Olvas(BeF,X,Y)
    Ciklus U=0-tól 9-ig
      Uj:=T[U,Y]+1; Ha Tu[U,X]<Uj akkor Tu[U,X]:=Uj
      Uj:=T[U,X]+1; Ha Tu[U,Y]<Uj akkor Tu[U,Y]:=Uj
    Ciklus vége
    Ciklus V=0-tól 9-ig
      Uj:=T[Y,V]+1; Ha Tu[X,V]<Uj akkor Tu[X,V]:=Uj
      Uj:=T[X,V]+1; Ha Tu[Y,V]<Uj akkor Tu[Y,V]:=Uj
    Ciklus vége
     T:=Tu
  Ciklus vége
  M:=0
  Ciklus X=0-tól 9-ig
    Ciklus Y=X-től 9-ig
      Ha M<T[x,y] akkor M:=T[x,y]
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Az i változó szerinti ciklus hatékonyabban is megvalósítható az alábbiak szerint:

```
  Ciklus i:=2-től N-ig
    Olvas(BeF,X,Y)
    Ha X>Y akkor U:=X; X:=Y;Y:=U
    Ciklus U=0-tól X-ig
      Uj:=T[U,X]+1; Ha Tu[U,Y]<Uj akkor Tu[U,Y]:=Uj
      Uj:=T[U,Y]+1; Ha Tu[U,X]<Uj akkor Tu[U,X]:=Uj
    Ciklus vége
    Ciklus U=X-től Y-ig
      Uj:=T[U,Y]+1; Ha Tu[X,U]<Uj akkor Tu[X,U]:=Uj
      Uj:=T[X,U]+1; Ha Tu[U,Y]<Uj akkor Tu[U,Y]:=Uj
    Ciklus vége
    Ciklus U=Y-tól X-ig
      Uj:=T[Y,U]+1; Ha Tu[X,U]<Uj akkor Tu[X,U]:=Uj
      Uj:=T[X,U]+1; Ha Tu[Y,U]<Uj akkor Tu[Y,U]:=Uj
    Ciklus vége
     T:=Tu
  Ciklus vége
```

2001. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mókusok (28 pont)

A. Az elején lesznek a diók — 3 pont
a végén pedig a mogyorók — 2 pont

B. Az i(=j) változó értéke az algoritmus végén vagy a diók száma, — 3 pont
vagy annál eggyel több — 3 pont
az előbbi, ha az első lyukban kezdetben dió volt, az utóbbi, ha nem — 3 pont

C. A helyén hagy minden diót, aminek a sorszáma kisebb vagy egyenlő volt a diók számánál — 2 pont
kivéve az elsőt, ha az dió volt — 2 pont
és minden mogyorót, aminek sorszáma nagyobb volt a diók számánál — 2 pont

D. A legjobb eset, ha elöl vannak a diók, mögötte a mogyorók — 2 pont
ekkor 2 mozgatás kell (az elsőt zsebre teszi, majd a zsebből visszateszi) — 2 pont
A legrosszabb eset, ha minden dió mogyoró helyén és minden mogyoró dió helyén van — 2 pont
ekkor N+1 mozgatás lesz (az elsőt kétszer kel mozgatni) — 2 pont

2. feladat: Szobanövény (17 pont)

A. 6 — 2 pont

B. 14 — 2 pont

C. 4 — 2 pont

D. 17 — 4 pont

E. 1,4,7,10, … — 7 pont

azaz minden harmadik évben (olyan számok, amik 3-mal osztva 1 maradékot adnak)

3. feladat: Hegymászó (24 pont)

Minden jó helyre sorolt pont 1 pontot ér. Nem adható pont azokra, amelyeket az egyes részfeladatokon belül több helyre is besoroltak.

A. 1. típus: 8,11,12 — 3 pont
2. típus: 2,3,5,6,7,9,10 — 7 pont
3. típus: 4,13 — 2 pont

B. 1/A típus: 8,12 — 2 pont
1/B típus: 11 — 1 pont
2/A típus: 5 — 1 pont
2/B típus: 3,7 — 2 pont
2/C típus: 9 — 1 pont
2/D típus: 10 — 1 pont
2/E típus: 2,6 — 2 pont
3/A típus: 4 — 1 pont
3/B típus: 13 — 1 pont

4. feladat. Osztálybuli (31 pont)

A. 1. 4 — 2 pont

2. 6 — 5 pont

B. 1. 500 vagy 600 vagy (650,700) — 3 pont

2. 5000 — 5 pont

C. 1. 3 — 3 pont

2. 7 — 5 pont

D. 1. A három időpontnak rendre az alábbi zárt intervallumokban kell lenni: — 3 pont
(66,99) (300,499) (650,700), vagy
(66,99) (500,500) (650,900), vagy
(100,100) (300,499) (650,700), vagy
(100,100) (500,600) (650,900)

2. A hét időpontnak (növekvő sorrendben) rendre az alábbi zárt intervallumokban kell lenni: — 5 pont

1. szám (601,1000)
2. és 3. szám (2000,3000) és (3001,3333) vagy
(2345,3000) és (3001,4000)
4. szám (4001,5000)
5. szám (5056,5156)
6. és 7. szám (5621,5999) és (6300,7000) vagy
(6000,6000) és (6300,9000)

Kilencedik-tizedik osztályosok

1. feladat: Gráf (20 pont)

A. Ha a j-edik pontba vezet él (legalább 1 pontból) — 3 pont

B. Ha az i-edik pontból nem indul él legalább 1 pontba — 3 pont

C. Ha az i-edik pontból mindenhova vezet él és — 3 pont

az i-edik pontba sehonnan nem vezet él — 3 pont

(az ilyen pontokat hívják a gráfban szuperforrásnak)

D. Csak egy ilyen pont lehet — 4 pont

Ha ugyanis belőle mindenhova vezet él, akkor nem lehet még egy olyan pont, ahova nem vezet él. — 4 pont

2. feladat: Bankár-algoritmus (20 pont)

Csoportonként adható pont, de csak akkor, ha az egyes csoportok sorszámait ilyen sorrendben adja meg és nem tesz közéjük más sorszámokat.

A jó sorrend: 1,2,3,4 — 4 pont

6,7 — 4 pont

5,9,10 — 4 pont

8,11,13 — 4 pont

12,14 — 4 pont

3. feladat: Lemezhibák (12 pont)

A. Egy foglalt (állományhoz tartozó) blokk a szabadok között is szerepel 3 pont

B. Egy nem foglalt (állományhoz nem tartozó) blokk nem szerepel a szabadok között 3 pont

C. `Ha T(i)>1 akkor HIBA3` 3 pont

D. `Ha S(i)>1 akkor HIBA4` 3 pont

4. feladat: Hegymászó (28 pont)

Minden jó csoport 5, illetve 1 pontot ér (akkor is, ha tettek bele nem oda illő pontot is).

A. 1. típus: 8,11,12,14,17 5 pont

2. típus: 2,3,5,6,7,9,10 5 pont

3. típus: 4,13,15,16,18,19 5 pont

B. 1/A típus: 16 1 pont

1/B típus: 4 1 pont

1/C típus: 15 1 pont

1/D típus: 19 1 pont

1/E típus: 13 1 pont

1/F típus: 18 1 pont

2/A típus: 5 1 pont

2/B típus: 3,7 1 pont

2/C típus: 9 1 pont

2/D típus: 10 1 pont

2/E típus: 2,6 1 pont

3/A típus: 8,12,17 1 pont

3/B típus: 11,14 1 pont

5. feladat: Kockarakás (20 pont)

A. Hapci: Az új kockát arra a toronyra teszi, amire rátehető és amelynek a tetején lévő kocka a legkisebb, ha nincs ilyen, akkor új toronyba 2 pont

Tudor: Az új kockát a legnagyobb felső kockájú toronyra teszi, ha lehet, ha nem lehet, akkor pedig új toronyba 2 pont

Kuka: Az új kockát vagy az első toronyra teszi, vagy új tornyot kezd vele 2 pont

B. Hapci<Tudor 2 pont

Tudor<Kuka 2 pont

C. Ha a kockák mérete monoton csökkenő 4 pont

D. Pl. 1 3 5 2 4 esetén Kuka 5 tornyot épít, Tudor négyet, Hapci pedig hármat

Hapci<Tudor, ha a legnagyobb nem az első tornyon van, amire egy kocka tehető 3 pont

Tudor<Kuka, ha Tudor tud az elsőtől különböző oszlopra is tenni kockát 3 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Gráf (18 pont)

A. Ha a j-edik pontba vezet él (legalább 1 pontból) 3 pont

B. Ha az i-edik pontból nem indul él legalább 1 pontba 3 pont

C. Ha az i-edik pontból mindenhova vezet él és 3 pont

az i-edik pontba sehonnan nem vezet él 3 pont

(az ilyen pontokat hívják a gráfban szuperforrásnak)

D. Csak egy ilyen pont lehet 3 pont

Ha ugyanis belőle mindenhova vezet él, akkor nem lehet még egy olyan pont, ahova nem vezet él. 3 pont

2. feladat: Háttértár kezelés (21 pont)

A. Következő-A: a legközelebbi sávra megy, ahol olvasáskérés van 3 pont

Következő-B: növekvő sorrendben halad az olvasáskéréseken, a legnagyobb sorszámúig, majd visszafordul és csökkenő sorrendben halad a legkisebb sorszámúig; a két irányt mindig a szélső kérésnél váltja 3 pont

Következő-C: növekvő sorrendben halad az olvasáskéréseken, a legnagyobb sorszámúig, majd a legkisebb sorszámúra lép vissza 3 pont

B. Következő-A: legtovább annak kell várni, aki a legmesszebb van a hívás sorozatban az éppen aktuális hívástól (minden lépés után újra vizsgálva) (ez vagy a legkisebb vagy a legnagyobb sorszámú) 2 pont

Következő-B: legtovább annak kell várni, aki haladási iránnyal ellentétben a legszélső (növekvő sorrendű haladáskor a legkisebb, csökkenőnél a legnagyobb) ha egyáltalán van ilyen; ha nincs, akkor pedig a haladási irányban legmesszebb levő
1+1 pont

Következő-C: legtovább annak kell várnia, akit éppen elhagyott az olvasófej (az aktuális sávnál kisebb sorszámúak közül a legnagyobb), az elhagyást ciklikusan kell érteni, azaz, ha mögötte nincs senki, akkor az előtte levők közül a legtávolabbi
1+1 pont

C. Következő-A: legkevesebbet az aktuális sávhoz legközelebbinek kell várni 2 pont

Következő-B: legkevesebbet a haladási irányban legközelebbinek kell várni, ha van ilyen; ha nincs, akkor az ellenkező irányban legközelebbinek 1+1 pont

Következő-C: legkevesebbet az aktuális sávnál nagyobbak közül legközelebbinek kell várnia, ha van ilyen; ha nincs, akkor a kisebbek közül a legnagyobbnak 1+1 pont

3. feladat: Szavak (26 pont)

A. 7 3 pont

B. A leghosszabb olyan tükörszó hossza, amelyet úgy kaphatunk, hogy S-ből betűket elhagyunk. 10 pont

C. 3 3 pont

D. Az S-be beszúrandó legkevesebb betű számát, aminek hatására S-ből tükörszó lesz. 10 pont

4. feladat: Autó rendezés (13 pont)

A. 5 3 pont

B. 8 4 pont

C. N/2 lefelé kerekítve. 6 pont

5. feladat: Jelek (22 pont)

Azok és csak azok a jelsorozatok, amelyek **0**-val kezdődő tükörszavak, és nem tartalmaznak két egymást követő **1**-est. 22 pont

Részekre bontva:

A. **0** 2 pont

B. **010** 4 pont

C. Ha **α** észlelt, akkor **0α0** is az. 8 pont

D. Ha **α** észlelt, akkor **01α10** is az. 8 pont

2001. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Automata (25 pont)

A megoldáshoz tárolnunk kell az automata pillanatnyi helyét (x,y), valamint irányát (irany). Arra kell még ügyelnünk, hogy az előre lépésnél a megfelelő irányba lépjünk.

```
Automata:
  x:=0; y:=0; irany:=0
  Ciklus amíg nem vége
    Be: betű
    Elágazás
      betű='E' esetén Elágazás
                        irany=0 esetén y:=y+1
                        irany=1 esetén x:=x+1
                        irany=2 esetén y:=y-1
                        irany=3 esetén x:=x-1
                      Elágazás vége
      betű='B' esetén Ha irany>0 akkor irany:=irany-1
                              különben irany:=3
      betű='J' esetén Ha irany<3 akkor irany:=irany+1
                              különben irany:=0
    Elágazás vége
  Ciklus vége
  Ki:'(',x,',',y,') '
  Elágazás
    irany=0 esetén Ki: 'észak'
    irany=1 esetén Ki: 'kelet'
    irany=2 esetén Ki: 'dél'
    irany=3 esetén Ki: 'nyugat'
  Elágazás vége
Eljárás vége.
```

2. feladat: Kolbász (24 pont)

A kol(N) tömbben tároljuk a kolbászok adatait. Minden időegységben a 0 értékű elemek szomszédai válnak nullává, s akkor ér véget a program futása, ha már nem marad elfogyasztandó kolbász.

```
Kolbász:
  ido:=0
  Ciklus
    db:=0; uj:='0'
    Ciklus i=2-től n-1-ig
      Ha kol(i)='0' akkor uj:=uj+'0'
      különben ha kol(i-1)='0' vagy kol(i+1)='0'
                  akkor uj:=uj+'0'
                különben uj:=uj+'1'; db:=db+1
    Ciklus vége
    uj:=uj+'0'; kol:=uj; ido:=ido+1
  amíg db>0
  Ciklus vége
Eljárás vége.
```

3. feladat: Számzár (26 pont)

Az első részfeladat megoldása a tárcsákon levő számok szorzata (a). A második részfeladat megoldása az elsőéből számítható úgy, hogy a szorzatból egyesével kihagyunk egy-egy elemet és ezeket utána összeadjuk (b). A harmadik részfeladatban minden tárcsapárra ki kell számolni, hogy azonos számjegyek esetén

```
Számzár:
  a:=1
  Ciklus i=1-től n-ig
    a:=a*t(i)
  Ciklus vége
  b:=0
  Ciklus i=1-től n-ig
    b:=b+a div t(i)
  Ciklus vége
  Ha b>a akkor b:=a
  c:=0
  Ciklus i=1-től n-1-ig
    Ciklus j=i+1-től n-ig
      Ha t(i)>t(j) akkor c:=c+a div t(i)
                   különben c:=c+a div t(j)
  Ha c>a akkor c:=a
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Jégtömb (20 pont)

Minden 1 értékű pontot 0-vá kell alakítani, ha legalább 2 nulla szomszédja volt. Az `Olvadás` eljárást addig kell újra és újra hívni, amíg az adott időegységben elolvadt jég mennyisége (`jég(idő)`) nem nulla.

```
Olvadás:
  jég(idő):=0
  Ciklus i=2-től n-1-ig
    Ciklus j=2-től m-1-ig
      Ha t(i,j)=1 akkor
        olvad(i,j):=(Szomszédszám(i,j)≥2)
        Ha nem olvad(i,j) akkor jég(idő):=jég(idő)+1
      különben olvad(i,j):=hamis
    Ciklus vége
  Ciklus vége
```

```
  Ciklus i=2-től n-1-ig
    Ciklus j=2-től m-1-ig
      Ha olvad(i,j) akkor t(i,j):=0
    Ciklus vége
  Ciklus vége
Eljárás vége.

Szomszédszám(i,j):
  s:=0
  Ha t(i-1,j)=0 akkor s:=s+1
  Ha t(i+1,j)=0 akkor s:=s+1
  Ha t(i,j-1)=0 akkor s:=s+1
  Ha t(i,j+1)=0 akkor s:=s+1
  Szomszedszam:=s
Eljárás vége.
```

2. feladat: Licit (20 pont)

Az i-edig igénylő az [A(i),B(i)] zárt intervallumba eső parcellákat igényli, és ezért Ft(i) forintot fizetne. Jelölje Opt(i) az 1..i parcellák eladásával elérhető legnagyobb bevételt!

Opt(0)=0. Ha i>0, akkor Opt(i) = max(Opt(i-1), Ft(j)+Opt(A(j)-1)), ahol B(j)=i.

Be(i)= j, ha Opt(i-1)<Ft(j)+Opt(A(j)-1)

DB(B(i))=a B(i)-re végződő igények száma.

Ig(B(i),k)=a B(i)-re végződő igények sorszámai, k=1-től Db(B(i))-ig.

```
Licit:
  Ciklus i=1-től M-ig
    Be: A(i),B(i),Ft(i))
    Inc(Db(B(i))); Ig(B(i),Db(B(i))):=i
  Ciklus vége
  Opt(0):=0; Be(0):=0
  Ciklus i=1-től N-ig
                {az i. parcelláig az optimális megoldás}
    Opt(i):=Opt(i-1); Be(i):=Be(i-1)
    Ciklus j=1-től Db(i)-ig {i-ben záruló igények közül a }
      k:=Ig(i,j)              {legtöbbet érő kiválasztása}
      Ha Opt(A(k)-1)+Ft(k)>Opt(i)
         akkor Opt(i):=Opt(A(k)-1)+Ft(k); Be(i):=k
    Ciklus vége
  Ciklus vége
  Ki: Opt(N); i:=N; MEG():=hamis
  Ciklus
    Meg(Be(i)):=igaz; i:=A(Be(i))-1
  amíg i>0 és Be(i)≠0
  Ciklus vége
Eljárás vége.
```

3. feladat: Térkép (20 pont)

Nagyon egyszerű megoldást kapunk, ha a térképet egy mátrixban tároljuk (van elég memória hozzá). Ekkor minden téglalap adatai beolvasásakor 1-esekkel kell feltölteni a mátrix megfelelő részét.

```
  Ciklus l=1-től k-ig
    Olvas(f,bfy,bfx,jay,jax)
    Ciklus i=bfy-tól jay-ig
      Ciklus j=bfx-től jax-ig
        t(i,j):=1
    Ciklus vége
  Ciklus vége
```

Ekkor a terület a mátrixban levő 1-esek száma, a kerület pedig az 1-esek szomszéd nullái száma.

```
Terület és kerület:
  terulet:=0
  Ciklus i=1-től n-ig
    Ciklus j=1-től m-ig
      Ha t(i,j)=1 akkor
        terulet:=terulet+1
        Ha i=1 vagy t(i-1,j)=0 akkor kerulet:=kerulet+1
        Ha i=n vagy t(i+1,j)=0 akkor kerulet:=kerulet+1
        Ha j=1 vagy t(i,j-1)=0 akkor kerulet:=kerulet+1
        Ha j=m vagy t(i,j+1)=0 akkor kerulet:=kerulet+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

4. feladat: Jelek (15 pont)

A feladat két részfeladat megoldásából áll: a jelek jöhettek-e az A égitestről, illetve jöhettek-e a B égitestről. Ha ezekre a kérdésekre tudjuk a választ, akkor ezekből előállítható a feladatban várt négyféle eredmény.

Akkor jöhet A-ról a jelsorozat, ha **0**, vagy **10**, vagy a szabály szerint visszavezethető **U0U1** vagy **U1U0** típusról az **U** típusúra.

```
Aról(X):
  e:=1; v:=hossz(X); Aról:=igaz
  Ciklus amíg 4≤v és Aról
    M:=(v-2) Div 2
    Ha nem (M>1 és páratlan(M) vagy X(M+1)=X(v)) akkor
                        {X(1..M)=X(M+2..v-1)?}
      i:=1; j:=M+2
      Ciklus amíg i≤M és X(i)=X(j)
        i:=i+1; j:=j+1
      Ciklus vége
      Ha i>M akkor v:=M különben Aról:=hamis
    különben Aról:=hamis
  Ciklus vége
  Aról:=v=1 és X(v)='0' vagy v=2 és X(1)='0' és X(2)='1'
Eljárás vége.
```

A második részfeladat egy helyes zárójelezés ellenőrzéséhez hasonló, teljesülnie kell minden egyes karakterre, hogy az előtte levő 0-k száma nem lehet kisebb z előtte levő 1-esek számánál, a sorozat végén pedig a 0-k és 1-esek számának egyenlőnek kell lennie.

```
Bről(X:):
  BRől:=igaz; N0:=0; i:=1
  Ciklus amíg i≤hossz(X) és N0≥0
    Ha X(i)='0' akkor N0:=N0+1 különben N0:=N0-1
    i:=i+1
  Ciklus vége
  BRől:=N0=0
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Jégtömb (20 pont)

Minden 1 értékű pontot 0-vá kell alakítani, ha legalább 3 nulla szomszédja volt. Az `Olvadás` eljárást addig kell újra és újra hívni, amíg az adott időegységben elolvadt jég mennyisége (`jég(idő)`) nem nulla.

```
Olvadás:
  jeg(ido):=0
  Ciklus l=2-től k-1-ig
    Ciklus i=2-től n-1-ig
      Ciklus j=2-től m-1-ig
        Ha t(i,j,l)=1 akkor
          Ha Szomszédszám(i,j,l)<3
             akkor jeg(ido):=jeg(ido)+1
          különben t(i,j,l):=2
      Ciklus vége
    Ciklus vége
  Ciklus vége
  Ciklus l=2-től k-1-ig
    Ciklus i=2-től n-1-ig
      Ciklus j=2-től m-1-ig
        Ha t(i,j,l)=2 akkor t(i,j,l):=0
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

```
Szomszédszám(i,j,l):
  s:=0
  Ha t(i-1,j,l)=0 akkor s:=s+1
  Ha t(i+1,j,l)=0 akkor s:=s+1
  Ha t(i,j-1,l)=0 akkor s:=s+1
  Ha t(i,j+1,l)=0 akkor s:=s+1
  Ha t(i,j,l+1)=0 akkor s:=s+1
  Ha t(i,j,l-1)=0 akkor s:=s+1
  Szomszédszám:=s
Függvény vége.
```

2. feladat: Licit (20 pont)

Az i-edig igénylő az [A(i),B(i)] zárt intervallumba eső parcellákat igényli, és ezért Ft(i) forintot fizetne. Ha B(i)<A(i), akkor az A(i)-től N-ig és 1-től B(i)-ig terjető parcellákat igényli.

Jelölje Opt(i,k) az i-edik parcellától legfeljebb j darab parcella eladásával elérhető legnagyobb bevételt!

Opt(i,0)=0. Ha k>0, akkor Opt(i,k) = max(Opt(i,k-1), Ft(j)+Opt(i,k-h)), ahol h=B(j)-A(j)+1, ha A(i)≤B(i), egyébként h=N-B(i)+A(i) és i+k-1=B(i) ha i+k-1<=N, egyébként i+k-1+N=B(i).

Be(i,k)= j, ha Opt(i,k-1)<Ft(j)+Opt(i,k-h)

DB(B(i))=a B(i)-re végződő igények száma

Ig(B(i),k)=a B(i)-re végződő igények sorszámai, k=1-től Db(B(i))-ig.

```
Licit:
  Ciklus i=1-től M-ig
    Be: A(i),B(i),Ft(i))
    Inc(Db(B(i))); Ig(B(i),Db(B(i))):=i
  Ciklus vége
  Ciklus i=1-től N-ig
    Opt(i,0):=0; Be(i,0):=0
  Ciklus vége
  Ciklus k=1-től N-ig
    Ciklus Kezd:=1-től N-ig
      Opti:=Opt(Kezd,k-1); Be(Kezd,k):=Be(Kezd,k-1)
      Veg:=Kezd+k-1
      Ha Veg>N akkor Veg:=Veg-N
      Ciklus j=1-től Db(Veg)-ig
        i:=Ig(Veg,j) {B(i)=Veg}
        h:=B(i)-A(i)+1; Ha h<=0 akkor h:=h+N
        Ha h≤k akkor
          Ha Opt(Kezd,k-h)+Ft(i)>Opt(Kezd,k) akkor
            Opt(Kezd,k):=Opt(Kezd,k-h)+Ft(i)
            Be(Kezd,k):=i
          Elágazás vége
        Elágazás vége
      Ciklus vége
    Ciklus vége
  Ciklus vége
  Opt:=0
  Ciklus i=1-től N-ig
    Ha Opti<Opt(i,N) akkor Opt:=Opt(i,N); Kezd:=i
  Ciklus vége;
  Ki: Opti
  Meg:=(hamis,…hamis); k:=N
  Ciklus
    Ciklus amíg Opt(Kezd,k-1)=Opt(Kezd,k)
      k:=k-1
    Ciklus vége
    i:=Be(Kezd,k); Meg(i):=igaz;
    h:=B(i)-A(i)+1; Ha h<=0 akkor h:=h+N
    k:=k-h
  amíg k>0 és Opt(Kezd,k)≠0
  Ciklus i=1-től M-ig
    Ha Meg(i) akkor Ki: i
  Ciklus vége
Eljárás vége.
```

3. feladat: Hálózat (15 pont)

Legyen T(x,y) az X-ből Y-ba vezető utak közül a legnagyobb sávszélességű. Ez azonos az x→y él sávszélességével, ha az útnak nincs közbülső pontja. Minden közbülső z pontra ismerhetjük az x→z és a z→y utak legnagyobb sávszélességét. Ha közülük a kisebbik értéke nagyobb, mint az x→y út eddig ismert legnagyobb sávszélessége, akkor a z-n keresztül vezető úton nagyobb a sávszélesség, tehát ezt kell tárolnunk T(x,y)-ban.

```
Számít(N,G,A,B,S):
  T:=G
  Ciklus z=1-től N-ig
    Ciklus x=1-től N-ig
      Ciklus y=1-től N-ig
        ujs:=T(x,z)              {ujs:=Min(T(x,z),T(z,y))}
        Ha T(z,y)<ujs akkor ujs:=T(z,y)
        Ha ujs>T(x,y) akkor T(x,y):=ujs; T(y,x):=ujs
      Ciklus vége
    Ciklus vége
  Ciklus vége
  S:=T(A,B)
Eljárás vége.
```

4. feladat: Gén (20 pont)

Első lépésként rendezzük a molekuladarabokat a bázissorrendjük szerint növekvő sorrendbe. Ezután keressük meg a k darab felbontás kezdőszekvenciáját, majd belőlük építsük fel a gént. A felépítéshez a szekvencia tömb első k elemébe tegyük a megtalált k darab kezdőszekvenciát, valamint jelöljük be, hogy ezek a szekvenciák már biztosan nem használhatók a gén más részeiben!

```
Összerakás:
  hossz:=0
  Ciklus i=1-től n-ig
    hossz:=hossz+hossz(gen(i)); szabad(i):=igaz
  Ciklus vége
  Ciklus i=1-től n-k+1-ig
    Ha Jókezdés(i) akkor
      Ciklus j=i-től i+k-1-ig
        szabad(j):=hamis; szekvencia(j-i+1):=gen(j)
      Ciklus vége
      Felépítés(1)
  Ciklus vége
Eljárás vége.
```

A Jókezdés függvény igaz értékű, ha k esetben igaz, hogy a j-edik molekuladarab bázissorrendje a j-1-edik molekuladarabbal kezdődik. A k féle darabolás ugyanis biztosan k első darabot eredményez, amelyek között nem lehet 2 egyforma. A rendezés miatt ezek biztosan egymás mögött helyezkednek el.

```
Jókezdés(i):
  j:=i+1
  Ciklus amíg j≤i+k-1 és gen(j-1)=gen(j,1..hossz(gen(j-1))
    j:=j+1
  Ciklus vége
  Jókezdés:=(j>i+k-1)
Függvény vége.
```

A k szekvencia közül mindig a legrövidebbet (az i sorszámút) próbáljuk bővíteni a többiekhez is illő következő darabbal! Ha sikerült így felépíteni a gént, akkor a feladat szerint meg is állhatnánk (pl. a Kiírás eljárásban lehetne egy HALT utasítás), de a mostani változatban az összes lehetséges felépítést megadjuk, azaz az illesztést visszavonjuk, és megyünk tovább.

```
Felépítés(i):
  Ciklus j=1-től n-ig
    Ha szabad(j) és Ráillik(i,szekvencia(i)+gen(j)) akkor
       ment:=szekvencia(i)
       szekvencia(i):=szekvencia(i)+gen(j)
       szabad(j):=hamis; min:=1
       Ciklus l=2-től k-ig
         Ha hossz(szekvencia(l))<hossz(szekvencia(min))
            akkor min:=l
       Ciklus vége
       Ha hossz(szekvencia(min))=hossz div k akkor Kiírás
       Felépítés(min)
       szekvencia(i):=ment; szabad(j):=igaz
  Ciklus vége
Eljárás vége.
```

Az új géndarab ráillik a legrövidebb szekvencia végére, ha az összes többihez is illeszkedik.

```
Ráillik(i,s):
  Ha i>1 akkor max:=1 különben max:=2
  Ciklus j=2-től k-ig
    Ha i≠j akkor
      Ha hossz(szekvencia(j))>hossz(szekvencia(max))
         akkor max:=j
  Ciklus vége
  j:=1
  Ciklus amíg j≤hossz(s) és j≤hossz(szekvencia(max))) és
              s(j)=szekvencia(max,j))
    j:=j+1
  Ciklus vége
  Ráillik:=nem(j≤hossz(s) és j≤hossz(szekvencia(max)))
Függvény vége.
```

2001. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Péntek (21 pont)

A feladat megoldásához szükségünk lesz a hónapok nevére, a hónapok napszámára, valamint a hét napjai nevére. Ezekhez egy-egy konstans tömböt használunk.

```
  hó: tömb(1..12,szöveg)=('január','február','március',
     'április','május','június','július','augusztus',
     'szeptember','október','november','december')
 nap: tömb(1..12,egész)=(31,28,31,30,31,30,31,31,30,31,30,31)
 hét: tömb(1..7,szöveg)=('szombat','vasárnap','hétfő',
      'kedd','szerda','csütörtök','péntek')
```

Szökőév esetén természetesen a február napszámát meg kell növelnünk eggyel! Ezután keressük meg a `hét` tömbben az év első napját! Számoljuk ki, hogy január 13. a hét tömb elemeihez képest hányadik napja a hétnek (a sorszámozást 0-tól kezdve)! Ha ennek a 7-tel osztási maradéka 0, akkor péntekre esik. Ezután adjuk hozzá az eddig kiszámolt értékhez a hónap napszámát, majd újra maradékvizsgálat következik, …!

```
Péntek13(év,első):
  Ha év mod 400=0 vagy év mod 100>0 és év mod 4=0
     akkor nap(2):=29
  j:=1
  Ciklus amíg első≠hét(j)
    j:=j+1
  Ciklus vége
  j:=j+5; h:=0
  Ciklus i=1-től 12-ig
    Ha (j+h) mod 7=0 akkor Ki: hó(i)
    h:=h+nap(i)
  Ciklus vége
Eljárás vége.
```

2. feladat: Hegy (27 pont)

A feladat megoldásához meg kell határozni az oda- és a visszaúton is az emelkedők kezdetét és végét, s közben közülük ki kell választani a leghosszabbat! Ha nincs emelkedő, akkor a leghosszabb emelkedő kezdete 0 marad.

```
Leghosszabb emelkedő(n,h):
  maxk:=0; maxv:=0; k:=0
  Ciklus i=2-től n-ig
    Ha k=0 és h(i)>h(i-1) akkor k:=i-1
    Ha k>0 és h(i)≤h(i-1) akkor
       Ha i-k>maxv-maxk+1 akkor maxk:=k; maxv:=i-1
       k:=0
    Elágazás vége
  Ciklus vége
  Ha k>0 akkor Ha n+1-k>maxv-maxk+1 akkor maxk:=k; maxv:=n
  k:=0
  Ciklus i=n-1-től 1-ig
    Ha k=0 és h(i)>h(i+1) akkor k:=i+1
    Ha k>0 és h(i)≤h(i+1) akkor
      Ha k-i>|maxk-maxv|+1 akkor maxk:=k; maxv:=i+1
      k:=0
    Elágazás vége
  Ciklus vége
  Ha k>0 akkor Ha k>|maxk-maxv|+1 akkor maxk:=k; maxv:=1
  Nincs:=(maxk=0)
Eljárás vége.
```

3. feladat: Bob (27 pont)

A megoldásban ki kell számolni, hogy melyik versenyző melyik poszton hányszor szerepelt! Ha az eltérés minden poszton legfeljebb 1, akkor az edző igazságos volt. Ebben az esetben kiírható, hogy ki lehet a következő versenyen *kormányos, második, harmadik*, illetve *indító*.

```
Bob(f,v):
  j:=1; igazság:=igaz; dbjo:=(f div 5)
  Ciklus amíg j≤4 és igazságos {posztonkénti ciklus}
    Ciklus i=1-től 5-ig
      db(i,j):=0
    Ciklus vége
```

```
    Ciklus i=1-től f-ig {számolás fordulónként}
      k:=1
      Ciklus amíg v(i,j)≠v(1,k)
        k:=k+1
      Ciklus vége
      db(k,j):=db(k,j)+1
    Ciklus vége
    k:=1
    Ciklus amíg k≤5 és (db(k,j)=dbjo vagy db(k,j)=dbjo+1)
      k:=k+1
    Ciklus vége
    igazságos:=(k>5); j:=j+1
  Ciklus vége
  Ha nem igazságos akkor Ki: 'Az edző nem volt igazságos!'
  különben Ki: 'Az edző igazságos volt!'
           Ciklus j=1-től 4-ig
             Ki: fun(j),' lehet:'
             Ciklus i=1-től 5-ig
               Ha db(i,j)=dbjo akkor Ki: v(1,i),' '
             Ciklus vége
           Ciklus vége
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Terem (16 pont)

Tároljuk az események kezdetét az `A(N)`, végét pedig a `B(N)` tömbben! A `Rendez` eljárás az események vége szerinti rendezettségi sorszámokat adja meg az `E(N)` tömbben.

Az eseményeket mohó stratégia szerint beosztjuk egy terembe. Amíg van még kimaradt esemény, addig újra alkalmazzuk a mohó stratégiát. A mohó stratégia alkalmazásainak száma, azaz a szükséges termek száma kerül a `T` változóba, a `Be(N)` tömb elemeibe pedig az egyes események teremsorszámát tesszük. Tegyük az egyes termek utolsó eseményeinek végét az U tömbbe!

A mohó stratégia lényege: az elsőnek kezdődő eseményt biztosan betehetjük az első terembe. Haladjunk tovább a kezdési idők szerint, s az eseményt osszuk be az első terembe, ahova befér. Ezután haladjunk tovább az eseményeken az utolsónak kezdődőig!

```
Terem(N,A,B):
  Rendez(N,B,E); T:=0; VanMég:=N; U():=0
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg A(E(i))>U(j)
      j:=j+1
    Ciklus vége
    Be(E(i)):=j; U(j):=B(E(i))
  Ciklus vége
Eljárás vége.
```

2. feladat: Túra (18 pont)

A folyók neve egy gráf pontjainak a neve lesz. Ezt a gráfot a beolvasás közben építhetjük fel az alábbiak szerint. A beolvasott nevet megkeressük a folyónevek táblázatában. Ha megtalálható benne, akkor megadjuk a sorszámát. Ha nincs meg, akkor felvesszük a legutolsó után, és ezt a sorszámot adjuk eredményként.

```
Keres(S,x):    {az S név keresése az FNev táblázatban}
  x:=0
  Ciklus
    x:=x+1
  amíg x≤M és FNev(x)≠S
  Ciklus vége
  Ha x>M akkor M:=M+1; FNev(x):=S
Eljárás vége.
```

Először számoljuk ki, hogy az első túra mely folyóra hány folyón keresztül juthat el! Ezután azt számoljuk meg, hogy a második túra hány folyón keresztül juthat olyan folyóra, ahol az első túra már járt!

Ha a második túra nem jut el olyan pontba, ahol az első túra is járt, akkor nem találkozhatnak (`Táv(x)=0` az eljárás végén). Ha eleve olyan pontban van, ahova az első túra eljutott, akkor a második bevárhatja az elsőt (`T=0` az eljárás végén). A harmadik esetben pedig azt tudjuk, hogy a két túra hol találkozhat (`x` az eljárás végén).

```
Túra():
  Táv():=0; T:=1; X:=F1; Tav(F1):=1
  Ciklus amíg Be(x)>0            {első túra}
    T:=T+1; x:=Be(x); Tav(x):=T
  Ciklus vége
  T:=0; X:=F2
  Ciklus amíg Be(x)>0 és Tav(x)=0      {második túra}
    T:=T+1; x:=Be(x)
  Ciklus vége
Eljárás vége.
```

3. feladat: Piramis (20 pont)

Minden (x,y) koordinátájú mezőre számítsuk ki az onnan induló leghosszabb út hosszát! A módszer: rekurzió memorizálással. Azért alkalmazható ez a módszer, mert bárhonnan indulva a lépkedési szabály szerint nem juthatunk olyan helyre, ahol már jártunk.

A megoldáshoz szükségünk lesz a négy szomszéd relatív koordinátáira:

```
  L: Tömb(1..4,rekord x,y:-1..+1) {a négy lehetséges lépés}
     =((x:-1;y:0),(x:0;y:1),(x:1;y:0),(x:0;y:-1));
```

```
MaxUt(x,y):
  Ha U(x,y)≠0 akkor MaxUt:=U(x,y) {(x,y)-ra már
                                    kiszámítottuk}
  különben Maxi:=0; Max4:=0
           Ciklus i=1-től 4-ig {négy lehetséges irány}
             xx:=x+L(i).x; yy:=y+L(i).y
             Ha P(x,y)+1=P(xx,yy) {arra lehet menni}
                akkor Max4:=MaxUt(xx,yy)
                      Ha Max4>Maxi akkor Maxi:=Max4
           Ciklus vége
           Maxi:=Maxi+1; U(x,y):=Maxi  {memorizálás}
           MaxUt:=Maxi
Eljárás vége.
```

```
Számít(N,M,P,U):
  Ciklus x=1-től N-ig      {inicializálás}
    P(x,0):=0; P(x,N+1):=0 {strázsa keret}
    Ciklus y=1-től N-ig
      P(0,y):=0; P(N+1,y):=0  {strázsa keret}
      U(x,y):=0;         {MaxUt(x,y)-t még nem számoltuk ki}
    Ciklus vége
  Ciklus vége
  Mego:=0
  Ciklus x=1-től N-ig
    Ha U(x,1)=0 akkor MaxUt(x,1)
    Ha U(x,1)>Mego akkor Mego:=U(x,1); Ix:=x; Iy:=1
    Ha U(x,N)=0 akkor MaxUt(x,N)
    Ha U(x,N)>Mego akkor Mego:=U(x,N); Ix:=x; Iy:=N
  Ciklus vége
  Ciklus y=1-től N-ig
    Ha U(1,y)=0 akkor MaxUt(1,y)
    Ha U(1,y)>Mego akkor Mego:=U(1,y); Ix:=1; Iy:=y
    Ha U(N,y)=0 akkor MaxUt(N,y)
    Ha U(N,y)>Mego akkor Mego:=U(N,y); Ix:=N; Iy:=y
  Ciklus vége
Eljárás vége.
```

4. feladat: Fazekas (21 pont)

Jelölje Opt(i) az első i-darab tárgy kiégetéséhez szükséges legrövidebb időt.

Opt(0)=0; Ha i>0, akkor

$$Opt(i) = \min\begin{cases} Opt(i-1) + Idő(i) \\ Opt(j-1) + \max Idő(j...i) \quad ahol \quad i-j+1 \le K \end{cases}$$

Egy optimális megoldás előállításához minden i-re tároljuk az E tömbben azt a j-t, amelyre a fenti minimum adódik. Ekkor az utolsó menetben az E(N)..N tárgyakat kell egy menetben a kemencében kiégetni. Ezt folytatva kapjuk meg a megoldást.

```
Fazekas(N,Ido):
  Opt(0):=0
  Ciklus i=1-től N-ig   {1..i tárgyakra optimális megoldás}
    Opti:=Opt(i-1)+Ido(i)  {az i. egyedül megy a kemencébe}
    Optj:=i; rido:=Ido(i); j:=i-1
    Ciklus amíg j>0 és i-j+1≤K {j..i egyszerre a kemencébe}
      Ha rido<Ido(j) akkor rido:=Ido(j)
      Ha Opt(j-1)+rido<Opti akkor Opti:=Opt(j-1)+rido
                                  Optj:=j
      j:=j-1
    Ciklus vége
    Opt(i):=Opti; E(i):=Optj  {E(i)..i megy egyszerre }
  Ciklus vége
  Ki: Opt(N); i:=N
  Ciklus amíg i≥1
    j:=E(i); E(j):=i {i..E(i) megy egy menetben}
    i:=j-1
  Ciklus cége
  i:=1
  Ciklus amíg i≤N
    Ki:i,E(i); i:=E(i)+1
  Ciklus cége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Folyók (10 pont)

A folyók neve egy gráf pontjainak a neve lesz. Ezt a gráfot a beolvasás közben építhetjük fel az alábbiak szerint. A beolvasott nevet megkeressük a folyónevek táblázatában. Ha megtalálható benne, akkor megadjuk a sorszámát. Ha nincs meg, akkor felvesszük a legutolsó után, és ezt a sorszámot adjuk eredményként.

```
Keres(S,x):     {az S név keresése az FNev táblázatban}
  x:=0
  Ciklus
    x:=x+1
  amíg x≤M és FNev(x)≠S
  Ciklus vége
  Ha x>M akkor M:=M+1; FNev(x):=S
Eljárás vége.
```

A `Be(N)` tömb elemeiből tudjuk azt, hogy az egyes folyók mely folyóba folynak bele. `IndN` az induló folyó neve. Legyen `Ki(x)=Igaz`⇔ ha a kijelölt folyó belefolyik x-be! Legyen `Bele(x)=Igaz`⇔ ha x belefolyik a kijelölt folyóba!

```
Számít(N,Be,Ki,Bele,IndN):
  Ciklus x=1-től FN-ig
    Bele(x):=hamis; Ki(x):=hamis
  Ciklus vége
  Keres(IndN,Ind); x:=Ind
  Ciklus amíg Be(x)≠0
    y:=Be(x); Ki(y):=igaz; x:=y
  Ciklus vége
  Ciklus x=1-től FN-ig {x bele folyik-e a kijelölt folyóba}
    y:=x
    Ciklus amíg Be(y)≠0 és y≠Ind
      y:=Be(y)
    Ciklus vége
    Ha y=Ind akkor Bele(x):=igaz
  Ciklus vége
  Bele(Ind):=hamis
Eljárás vége.
```

2. feladat: Kemence (15 pont)

Jelölje Opt(i) az első i-darab tárgy kiégetéséhez szükséges legrövidebb időt.

Opt(0)=0; Ha i>0, akkor

$$Opt(i) = \min\begin{cases} Opt(i-1) + Idő(i) \\ Opt(j-1) + \max Idő(j...i) \quad ahol \quad szummaSúly(j...i) \le K \end{cases}$$

Egy optimális megoldás előállításához minden i-re tároljuk az E tömbben azt a j-t, amelyre a fenti minimum adódik. Ekkor az utolsó menetben az E(N)..N tárgyakat kell egy menetben a kemencében kiégetni. Ezt folytatva kapjuk meg a megoldást.

```
Dinamikus:
  Opt(0):=0
  Ciklus i=1-től N-ig {1..i tárgyakra optimális megoldás}
    Opti:=Opt(i-1)+S(i).ido {i. egyedül megy a kemencébe}
    Optj:=i; rido:=S(i).ido; E(i):=i
    rsuly:=S(i).suly; j:=i-1
    Ciklus amíg (j>0) és (rsuly+S(j).suly≤K) {j..i együtt?}
      Ha rido<S(j).ido akkor rido:=S(j).ido
      rsuly:=rsuly+S(j).suly
      Ha Opt(j-1)+rido<Opti akkor Opti:=Opt(j-1)+rido
                                  Optj:=j
      j:=j-1
    Ciklus vége
    Opt(i):=Opti; E(i):=Optj {E(i)..i megy egy menetben}
  Ciklus vége
  Ki: Opt(N); i:=N
  Ciklus amíg i≥1
    j:=E(i); E(j):=i {i..E(i) megy egy menetben}
    i:=j-1
  Ciklus cége
  i:=1
  Ciklus amíg i≤N
    Ki:i,E(i); i:=E(i)+1
  Ciklus cége
Eljárás vége.
```

3. feladat: Hegység (20 pont)

Határozzuk meg minden (x,y) helyről induló leghosszabb út hosszát!. A módszer: rekurzió memorizálással. Azért alkalmazható ez a módszer, mert bárhonnan indulva a lépkedési szabály szerint nem juthatunk olyan helyre, ahol már jártunk.

A megoldáshoz szükségünk lesz a négy szomszéd relatív koordinátáira:

```
  L: Tömb(1..4,rekord x,y:-1..+1) {a négy lehetséges lépés}
     =((x:-1;y:0),(x:0;y:1),(x:1;y:0),(x:0;y:-1));
```

```
MaxUt(x,y):
  Ha U(x,y)≠0 akkor MaxUt:=U(x,y)
  különben Maxi:=0; Max4:=0
           Ciklus i=1-től 4-ig {négy lehetséges irány}
             xx:=x+L(i).x; yy:=y+L(i).y
             Ha T(xx,yy)>0 és T(x,y)<T(xx,yy)
                akkor Max4:=MaxUt(xx,yy)
                      Ha Max4>Maxi akkor Maxi:=Max4
           Ciklus vége
           Maxi:=Maxi+1; U(x,y):=Maxi  {memorizálás}
           MaxUt:=Maxi
Eljárás vége.
```

```
Számít(N,M,P,U):
  Ciklus x=1-től N-ig      {inicializálás}
    P(x,0):=0; P(x,N+1):=0 {strázsa keret}
    Ciklus y=1-től N-ig
      P(0,y):=0; P(N+1,y):=0  {strázsa keret}
      U(x,y):=0        {MaxUt(x,y)-t még nem számoltuk ki}
    Ciklus vége
  Ciklus vége
```

```
  Mego:=0
  Ciklus x=1-től N-ig
    Ha U(x,1)=0 akkor MaxUt(x,1)
    Ha U(x,1)>Mego akkor Mego:=U(x,1); Ix:=x; Iy:=1
    Ha U(x,N)=0 akkor MaxUt(x,N)
    Ha U(x,N)>Mego akkor Mego:=U(x,N); Ix:=x; Iy:=N
  Ciklus vége
  Ciklus y=1-től N-ig
    Ha U(1,y)=0 akkor MaxUt(1,y)
    Ha U(1,y)>Mego akkor Mego:=U(1,y); Ix:=1; Iy:=y
    Ha U(N,y)=0 akkor MaxUt(N,y)
    Ha U(N,y)>Mego akkor Mego:=U(N,y); Ix:=N; Iy:=y
  Ciklus vége
Eljárás vége.
```

Vegyük észre, hogy a feladat megoldása szinte szó szerint azonos az eggyel kisebb korcsoport PIRAMIS feladata megoldásával. Z egyetlen különbség a vastagon szedett rész: itt nemcsak egyetlen lépéssel lehet magasabbra lépni, hanem akármennyivel.

4. feladat: Vírus (30 pont)

Vegyük észre, hogy a víruskifejezés megadható rekurzív definícióval. Minden szabályos víruskufejezés tagok egymás után írt sorozata. Minden tag vagy egy betű, vagy a '-' karakter, vagy kifejezéseknek a '|' karakterrel elválasztott sorozata, amely a '[' nyitó és a ']' záró zárójelek között áll.

A kifejezés és a tag elemzésére készítsünk rekurzív függvényeljárást. A Kifejezés függvény bemenete a vizsgált vírus karaktersorozat pozícióit tartalmazó P halmaz. A kimenete legyen az összes olyan j index, amelyre teljesül, hogy van olyan i∈P, hogy a V[i..j] karaktersorozat megfelel a hívással elemzett víruskifejezésnek.

Ekkor egy V karaktersorozat akkor és csak akkor tartozik a K víruskifejezés által jelölt vírustörzsbe, ha hossz(V) ∈Kifejezés({0}). Ha ez nem teljesül, akkor azt a legnagyobb i indexet kell kiszámítani, amelyre teljesül, hogy a V[1..i] részsorozat megfelel a K kifejezés valamely kezdőszeletének. Ezt úgy számíthatjuk ki, hogy minden Tag hívás után meghatározzuk a kimeneti halmaz legnagyobb elemét, és ha ez nagyobb, mint az eddigi legynagyobb, akkor feljegyezzük.

```
Számít:
  Be: K {a víruskifejezés beolvasása}
  Be: M  {a vizsgálandó vírusok száma;}
  Ciklus i:=1-tól M-ig
    Be: V {vírus beolvasása};
    Vh:=V-hossza; P:={0}; maxi:=0
    Q:=Kifejezes(P)
    Ha Vh∈Q akkor Ki: „VIRUS” különben Ki: maxi
  Ciklus vége
Eljárás vége.
```

```
Kifejezés(P):
  Ciklus
    Q:=Tag(P); P:=Q
  amíg K(poz)∉{'|',']','.'}
  Ciklus vége
  Kifejezés:=Q
Eljárás vége.
```

```
Tag(P):
  Ha K(poz)∈{'A'..'Z'} akkor
    Q:=∅
    Ciklus x=1-tól Vh-ig
      Ha x-1∈P és V(x)=K(poz) akkor Q:=Q∪{x}
    Ciklus vége
    poz:=poz+1
  különben ha K(poz)='-' akkor Q:=P
  különben {K(poz)='['}
    poz:=poz+1; Q:=Kifejezés(P)
    Ciklus amíg K(poz)='|'
      poz:=poz+1; Q1:=Kifejezés(P); Q:=Q∪Q1
    Ciklus vége
    poz:=poz+1   {] átlépése}
  Elágazás vége
  Tag:=Q
  Ciklus i:=1-től Vh-ig
    Ha i∈Q és i>maxi akkor maxi:=i
  Ciklus vége
Eljárás vége.
```

2002. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Válogatós (26 pont)

A. (Nóra, Erika, János) 4 pont

Részpontszámok lépésenként:

(-, János, Nóra)	1 pont
(Nóra, János, -)	1 pont
(Nóra, -, János)	1 pont
(Nóra, Erika, János)	1 pont

B. (Piri, Emese, Móni, Enikő, Eszter, Anna, Jakab, András, László, Béla, Péter) 8 pont

Részpontszámok lépésenként:

(-, Emese, Béla, Enikő, András, Jakab, Eszter, Móni, László, Piri, Péter)	1 pont
(Piri, Emese, Béla, Enikő, András, Jakab, Eszter, Móni, László, -, Péter)	1 pont
(Piri, Emese, -, Enikő, András, Jakab, Eszter, Móni, László, Béla, Péter)	1 pont
(Piri, Emese, Móni, Enikő, András, Jakab, Eszter, -, László, Béla, Péter)	1 pont
(Piri, Emese, Móni, Enikő, -,Jakab, Eszter, András, László, Béla, Péter)	1 pont
(Piri, Emese, Móni, Enikő, Eszter, Jakab, -, András, László, Béla, Péter)	1 pont
(Piri, Emese, Móni, Enikő, Eszter, -, Jakab, András, László, Béla, Péter)	1 pont
(Piri, Emese, Móni, Enikő, Eszter, Anna, Jakab, András, László, Béla, Péter)	1 pont

C. Csak az elsőnek kell mozognia, 2 pont
ha a második széktől kezdve csak fiúk ülnek a sorban. 2 pont
Az első széken lány is, fiú is ülhet 2 pont

D. Kezdetben minden lány fiú helyén ül és 3 pont
minden fiú lány helyén ül, 3 pont
az első széket kivéve (ahol lány is, fiú is ülhet). 2 pont

Más megfogalmazásban: Ha a székek száma páros (2*K alakú), akkor
az első széken mindegy ki ül, 1 pont
utána K-1 fiú következik, 2 pont
utána K lány következik 2 pont

Ha a székek száma páratlan (2*K+1 alakú), akkor
az első széken mindegy ki ül, 1 pont
utána K fiú következik, 1 pont
utána K lány következik 1 pont

2. feladat: Kockapóker (30 pont)

?1 póker	2 pont
?2 két pár	2 pont
?3 terc	2 pont
?4 terc	2 pont
?5 egy pár	2 pont
?6 terc	2 pont
?7 terc	2 pont
?8 egy pár	2 pont
?9 két pár	2 pont
?10 egy pár	2 pont
?11 két pár	2 pont
?12 egy pár	2 pont
?13 egy pár	2 pont
?14 egy pár	2 pont

?15 semmi 2 pont

3. feladat: Címletezés (29 pont)

A. i) DB=(0,1,0,1,1,0) 3 pont
ii) DB=(3,0,0,1,0,1) 3 pont
iii) DB=(1,0,0,1,1,0) 3 pont
iv) DB=(3,0,0,1,0,1) 3 pont

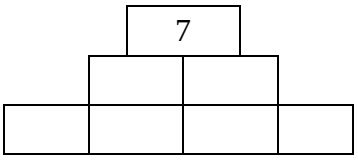
B. Hibás: i), ii), iv), jó: iii) 2+2+2+2 pont

C. i) P=12, algoritmus=(2,0,0,1,0,0), helyes=(0,0,2,0,0,0) 1+1+1 pont
ii) P=8, algoritmus=(2,0,1,0,0,0), helyes=(0,2,0,0,0,0) 1+1+1 pont
iv) P=10, algoritmus=(3,0,1,0,0,0), helyes=(0,2,0,0,0,0) 1+1+1 pont

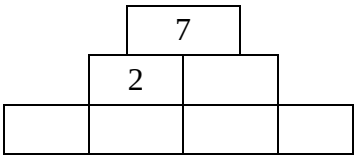
4. feladat: Piramis (15 pont)

A. lépésenként 1-1 pont 7 pont

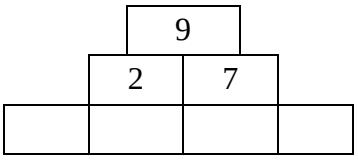
1.



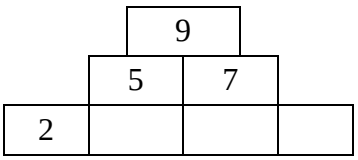
2.



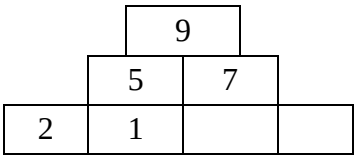
3.



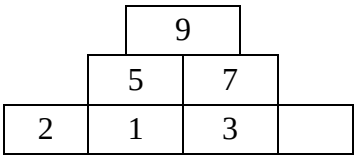
4.



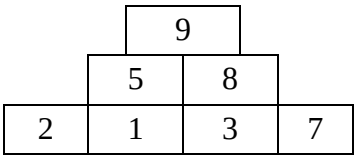
5.



6.



7.



B. A legnagyobb szám lesz a piramis tetején 2 pont

C. Az I sorszámú kisebb a fölötte levőnél 2 pont
vagy egyenlő vele 1 pont

Az I sorszámú nagyobb az alatta levőknél 2 pont
vagy egyenlő velük 1 pont

Kilencedik-tizedik osztályosok

1. feladat: Válogatós (24 pont)

A. `(Nóra, Erika, János)` 4 pont

Részpontszámok lépésenként:

`(-, János, Nóra)` 1 pont
`(Nóra, János, -)` 1 pont
`(Nóra, -, János)` 1 pont
(Nóra, Erika, János) 1 pont

B. (Piri, Emese, Móni, Enikő, Eszter, Anna, Jakab, András, László, Béla, Péter) 8 pont

Részpontszámok lépésenként:

(-, Emese, Béla, Enikő, András, Jakab, Eszter, Móni, László, Piri, Péter) 1 pont
(Piri, Emese, Béla, Enikő, András, Jakab, Eszter, Móni, László, -, Péter) 1 pont
(Piri, Emese, -, Enikő, András, Jakab, Eszter, Móni, László, Béla, Péter) 1 pont
(Piri, Emese, Móni, Enikő, András, Jakab, Eszter, -, László, Béla, Péter) 1 pont
(Piri, Emese, Móni, Enikő, -,Jakab, Eszter, András, László, Béla, Péter) 1 pont
(Piri, Emese, Móni, Enikő, Eszter, Jakab, -, András, László, Béla, Péter) 1 pont
(Piri, Emese, Móni, Enikő, Eszter, -, Jakab, András, László, Béla, Péter) 1 pont
(Piri, Emese, Móni, Enikő, Eszter, Anna, Jakab, András, László, Béla, Péter) 1 pont

C. Csak az elsőnek kell mozognia, 2 pont
ha a második széktől kezdve csak fiúk ülnek a sorban. 2 pont
Az első széken lány is, fiú is ülhet 2 pont

D. Kezdetben minden lány fiú helyén ül és 2 pont
minden fiú lány helyén ül, 2 pont
az első széket kivéve (ahol lány is, fiú is ülhet). 2 pont

Más megfogalmazásban:

Ha a székek száma páros (2*K alakú), akkor
az első széken mindegy ki ül, 1 pont
utána K-1 fiú következik, 1 pont
utána K lány következik 1 pont

Ha a székek száma páratlan (2*K+1 alakú), akkor
az első széken mindegy ki ül, 1 pont
utána K fiú következik, 1 pont
utána K lány következik 1 pont

2. feladat: Szakaszrajzolás (12 pont)

A program csak x2>x1 esetén működik jól. 3 pont

Nullával osztana x1=x2 esetén. 2 pont

Emiatt az eljárás nem tud függőleges vonalat húzni. 2 pont

Nem rajzol semmit, ha x2<x1. 2 pont

Ha a szakasz túl meredek (több mint 45 fok), akkor nem rajzol folytonos vonalat. 3 pont

3. feladat: Rekurzió (23 pont)

A) FEDCBA 3 pont

Megfordítja a szöveg karaktereinek sorrendjét 5 pont

B) BDFECA 3 pont

A páros sorszámú karakterek sorrendjét megtartja, és előre hozza őket. 3 pont

A páratlan sorszámúakat a végére teszi, fordított sorrendben. 3 pont

C) EDCBA 3 pont

D) BDECA 3 pont

4. feladat: Mit csinál? (20 pont)

A. B(i) = az A(i)-nál kisebb elemek száma az A vektorban + 1 5 pont

B. C(i) = az i-edik helyre kerülő elem, ha C(i) nemnegatív 3 pont

C(i) = -1, ha az i-edik helyre olyan elem kerülne, amely már szerepel C-ben 3 pont

C. C = az A vektor elemei növekvő sorrendben 3 pont

D. Az A vektort rendezve teszi a C vektorba 3 pont

növekvő sorrendbe rendez 3 pont

5. feladat: Kódolás (22 pont)

A1. A `Csoportolvasás` eljárás nem figyeli az állomány végét 2 pont

A2. Az utolsó karakter nem kerül bele az eredménybe 2 pont

A3. Ha több mint 255 azonos karakter van egymás mellett, akkor a darabszám nem vagy hibásan alakítható át ascii-kóddá 4 pont

B1. `Ciklus amíg k=kar és nem filevége(X)` 2 pont

B2. Az állomány lezárása előtt ki kell írni Z-be a `k`-ban levő karaktert 2 pont

B3. `Ciklus amíg k=kar és nem filevége(X) és db<255` 2 pont

C. Ha a karakter egymagában áll, illetve ha csak 2 egyforma karakter jön egymás után, akkor is 3 karakterrel helyettesít 4 pont

D. Az `Ír(Z,⊕+Karakter(db)+kar)` utasítást egy elágazásba kell tenni, amit csak `db>2` esetén hajtunk végre, egyébként pedig ki kell írni `db` darab `kar` karaktert 4 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rendezés (20 pont)

A) `Egyik`: A `B` változó az `i` ... `N-1` indexű elemek közül a legkisebb indexe 3 pont

`Másik`: A `B` változó az `i` ... `N-i-1` indexű elemek közül a legkisebb indexe 3 pont

A `D` változó az `i` .... `N-i-1` indexű elemek közül a legnagyobb indexe 3 pont

B. `Egyik`: $N*(N-1)/2 = N^2/2 - N/2$ 3 pont

`Másik`: $N/2+$ {*} 2 pont

$(N/2)*(N-2)/2 +$ {**} 2 pont

$(N/2)*(N-2)/2 =$ {***} 2 pont

$= N/2+(N/2)*(N-2) = N/2+N^2/2-N = N^2/2-N/2$

C. A fentiekből látszik, hogy az összehasonlítások száma egyforma. 2 pont

2. feladat: Rekurzió (20 pont)

A. `FBDCEA` 2 pont

Az elejéről, illetve a végéről befelé haladva a páratlan sorszámúakat felcseréli egymással (az elsőt az utolsóval, a harmadikat a hátulról harmadikkal stb.), míg a páros helyen lévők a helyükön maradnak (a második és az utolsó előtti, a negyedik és a hátulról negyedik...). 2 pont

B. `AECDBF` 2 pont

Az elejéről, illetve a végéről befelé haladva a páros sorszámúakat felcseréli egymással (a másodikat és az utolsó előttit, a negyediket és a hátulról negyediket stb.), míg a páratlan helyen lévők a helyükön maradnak (az első és az utolsó, a harmadik és a hátulról harmadik stb.). 2 pont

C. `BJDHFEGCIA` 2 pont

A páros sorszámúakat előre hozza úgy, hogy (elölről nézve) elölről és hátulról váltogatja a karaktereket, 2 pont

a páratlan sorszámúakat pedig a végére viszi úgy, hogy (hátulról nézve) ugyancsak elölről és hátulról váltogatja a karaktereket. 2 pont

D. `JBHDFECGAI` 2 pont

A páros sorszámúakat előre hozza úgy, hogy (elölről nézve) hátulról és elölről váltogatja a karaktereket, 2 pont

a páratlan sorszámúakat pedig a végére viszi úgy, hogy (hátulról nézve) ugyancsak hátulról és elölről váltogatja a karaktereket. 2 pont

3. feladat: Szövegkeresés (18 pont)

A. A `dh` változóban a d^{m-1} `mod q` érték van 2 pont

Ha az `m` nagy (az `sminta` hosszú), akkor a ciklusban túlcsordulás lehetne, ha nem csak a maradékot tárolnánk 2 pont

B. Az `s1` az `sminta` karaktereiből előállított kód 2 pont

Az `s2` az `s` 1..m. karaktereiből előállított kód 2 pont

C. Az `i`-edik karaktert kiveszi a kódból 2 pont

A többi karakter kódját szorozza `d`-vel (lépteti a kódban) 2 pont

Az `i+m`-edik karaktert hozzáveszi a kódhoz 2 pont

D. Ha a kód nagyobb `q`-nál (hosszú az `sminta`), akkor a `mod` művelet miatt két különböző karaktersorozathoz is ugyanazt a kódot rendeljük; az utolsó ciklus a kódok azonossága esetén áll le, ekkor az értékek különbözhetnek 4 pont

4. feladat: Adatbázis (22 pont)

A1:
```
SELECT kód, létszám                                  2 pont
  FROM Osztályok                                     2 pont
  WHERE létszám < 30                                 2 pont
```

A2:
```
SELECT név, létszám                                  2 pont
  FROM Osztályfőnökök, Osztályok                     2 pont
  WHERE osztály=kód AND város="Debrecen"             2 pont
```

B: Kiírt nevek: `Kovács Pál`, `Szabó János` 2+2 pont
Minden hibáért egy pont levonás jár a 4-ből!

C.
```
SELECT név                                           1 pont
  FROM Osztályfőnökök                                1 pont
  WHERE azon=(SELECT vezető                          1+1 pont
                FROM Osztályfőnökök                  1 pont
                WHERE szak="Magyar")                 1 pont
```

5. feladat: Prioritási sor (20 pont)

A. Az első tömbelemben lesz a legnagyobb prioritású elem 3 pont

B. `K(i).pr`≥`K(2*i).pr` és `K(i).pr`≥`K(2*i+1).pr` minden `i`-re `hossz div 2`-ig 3+3 pont

C. A `Sorba` eljárás `hossz div 2`-től indul és felezget, amíg 1 alá nem érünk, így kb. $\log_2$(`hossz`) lesz a maximális lépésszám. 3 pont

A `Sorból` eljárás 1-től indul és kétszerez, amíg `hossz div 2` fölé nem érünk, így kb. $\log_2$(`hossz`) lesz a maximális lépésszám 3 pont

D. (A,5) 1 pont

(A,5), (B,3) 1 pont

(C,7), (B,3), (A,5) 1 pont

(C,7), (B,3), (A,5), (D,2) 1 pont

(C,7), (E,4), (A,5), (D,2), (B,3) 1 pont

(A,5), (E,4), (B,3), (D,2) 1 pont

2002. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mókus (23 pont)

A megoldásban végignézzük a mókusokat (`k(m)` tömb), s minden olyan kőnél számolunk egyet, ahova léphetnek (`DB`). Az eljárás végén a 0 értékű számlálók mutatják azokat a köveket, ahova nem lépett mókus, a maximális értékű számláló pedig azt, ahova a legtöbb mókus lépett.

```
Mókus(n,m,k,db):
  db():=0
  Ciklus i=1-től m-ig
    j:=k(i)
    Ciklus amíg j≤n
      db(j):=db(j)+1; j:=j+k(i)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

2. feladat: Torlódás (25 pont)

Az `s` szöveges változóban levő szövegben a mássalhangzók helyét vizsgáljuk. Minden mássalhangzótól (1-, 2- vagy 3-jegyű) haladunk a következő magánhangzóig, számolva a közben átlépett mássalhangzókat. A leghosszabb ilyen szakasz kezdete (`h`) és hossza (`hdb`) lesz a megoldás.

```
Torlódás(s):
  hdb:=0; h:=0; i:=1
  Ciklus amíg i≤hossz(s)
    Ha mássalhangzó(i) akkor
      j:=i; jdb:=1
      Ha s(i)+s(i+1)+s(i+2)='dzs' akkor i:=i+3
      különben ha kettősbetű(i) akkor i:=i+2
      különben i:=i+1
      Ciklus amíg i≤hossz(s) és mássalhangzó(i)
        Ha s(i)+s(i+1)+s(i+2)='dzs' akkor i:=i+3
           különben ha kettősbetű(i) akkor i:=i+2
           különben i:=i+1
        jdb:=jdb+1
      Ciklus vége
      Ha jdb>hdb akkor h:=j; hdb:=jdb
    különben i:=i+1
  Ciklus vége
Eljárás vége.
```

Mivel a nem egyértelműen elemezhető szavakat kiszűrtük, csak azt kell megvizsgálnunk, hogy mit tesz az algoritmusunk a hosszú kétjegyű mássalhangzókkal. Vegyük például az **ssz**-t! Nála az első **s** betűre számolunk egyet, majd az **sz**-re még egyet, azaz helyesen két mássalhangzónak számoljuk (bár nem két **sz** betűt számoltunk).

3. feladat: „Egyes” számok (27 pont)

Első lépésként adjuk össze az S tömbben található számjegyeket! Ha egyjegyű számot kapunk, akkor készen vagyunk. Ha a szám nem egyjegyű, akkor mindaddig ismételjük az alábbiakat, amíg nem kapunk egyjegyű számot: az adott szám számjegyeit adjuk össze, majd vegyük ezt következő számnak.

```
Egyes számok(N,S)
  j:=0
  Ciklus i=1-től N-ig
    j:=j+S(i)
  Ciklus vége
  Ciklus amíg j>9
    k:=0
    Ciklus amíg j>0
      k:=k+j mod 10; j:=j div 10
    Ciklus vége
    j:=k
  Ciklus vége
  Ha j=1 akkor KI: 'EGYES' különben Ki: 'NEM EGYES'
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Tipp (15 pont)

Első lépésként számoljuk meg, hogy melyik tippet hányszor adták (az `N` elemű `t` tömbben hányszor fordul elő), majd a darabszámokat tartalmazó tömbnek határozzuk meg a maximumát!

A maximális tippelők közül azokat kell kiválogatni, akik elsők voltak annak a számnak a tippelésében, méghozzá a tipp értékel szerint növekvő sorrendben.

```
Tipp(N,db,t)
  db():=0
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg t(j)≠t(i)
      j:=j+1
    Ciklus vége
    db(j):=db(j)+1
  Ciklus vége
  max:=1
  Ciklus i=2-től N-ig
    Ha db(i)>db(max) akkor max:=i
  Ciklus vége
  j:=0
  Ciklus i=1-től N-ig
    Ha db(i)=db(max) akkor
       j:=j+1; t(j):=t(i); k:=j-1
       Ciklus amíg k>0 és t(k)>t(k+1)
         tt:=t(k+1); t(k+1):=t(k); t(k):=tt; s(k+1):=s(k)
         k:=k-1
       Ciklus vége
       t(k+1):=t(i); s(k+1):=i
  Ciklus vége
Eljárás vége.
```

2. feladat: Ütemezés (20 pont)

Minden munkát próbáljunk beosztani a határideje előtti legutolsó szabad napra! Belátható, hogy ezzel a mohó megoldással megkapjuk az optimális megoldást. Ha ugyanis egy munkát korábbra osztanánk be, akkor a további munkákat vagy ugyanúgy be tudnánk osztani, vagy pedig valamelyiket az előre hozott munka miatt nem lehet sehova beosztani.

```
Ütemezés:
  Be: N
  Beoszt():=0; M:=0;    {a beosztott munkák száma}
  Ciklus i=1-től N-ig
    Be: h            {az i. munka határideje h}
    nap:=h
    Ciklus amíg nap>0 és Beoszt(nap)>0
      nap:=nap-1
    Ciklus vége
    Ha nap≠0 akkor Beoszt(nap):=i; M:=M+1
  Ciklus vége
  Ki: M
  Ciklus i=1-től MaxNap-ig
    Ha Beoszt(i)≠0 akkor Ki: Beoszt(i),i
  Ciklus vége
Eljárás vége.
```

3. feladat: Üvegválogatás (20 pont)

Minden lehetséges megoldás megadható egy $<i_1,...,i_k>$ indexsorozattal, ahol a j-edik szám annak a ládának a sorszáma, amelyet a j-edik fajta számára kijelölünk. Nyilvánvalóan, hogy az indexeknek páronként különbözőnek kell lenniük. Olyan $<i_1,...,i_k>$ sorozatot kell keresni, amelyre az Osszeg($L(i_j,j)$:j=1..k) maximális, ahol L(i,j) az i-edik ládában lévő j-edik fajta üvegek száma! A keresést viszszalépéses stratégiával végezhetjük.

```
Válogatás(N,K,L):
  Opt:=0 {az optimális megoldás értéke}
  Rész:=0 {a megoldáskezdeményhez tartozó összeg}
  NemVolt():=(Igaz, ..., Igaz)
  Keres(0)
  Ki: Total-Opt, OptX
Eljárás vége.
```

A Keres eljárás a megoldástér rekurzív bejárását végzi. Keres(j) hívás előtt teljesül, hogy az X megoldáskezdemény első j elemét már kiválasztottuk, az eljáráshívás a j+1-edik elemet állítja be. Az X vektor globális az eljárásra nézve. Az Opt változó tartalmazza az eddig talált legjobb megoldás célfüggvényértékét. A Rész változó értéke az Osszeg($L(i_u,u)$:u=1..j), ha X=($i_1,...,i_j$). OptX egy optimális megoldás vektor.

```
Keres(j):
  Ciklus i=1-től N-ig
    Ha NemVolt(i) akkor
      X(j+1):=i
      Rész:=Rész+L(i,j+1); NemVolt(i):=hamis     {bejegyez}
      Ha j+1=K és Rész>Opt akkor {jobb megoldást találtunk}
        Opt:=Rész; OptX:=X
      Ha j<K akkor Keres(j+1)              {rekurzív hívás}
      Rész:=Rész-L(i,j+1); NemVolt(i):=igaz   {visszaállít}
  Ciklus vége
Eljárás vége.
```

4. feladat: Tükörszó (20 pont)

Az `N` karakterből álló `S` szót kell felbontani tükörszavakra! Legyen `R(i)` = az `S(1..i)` szó legkevesebb tükörszóra bontásának száma! Ekkor a feladat megoldása az `R(N)` érték kiszámolása.

A megoldásban `T(i,j)` akkor és csak akkor legyen igaz értékű, ha `S(i..j)` tükörszó. Nyilvánvaló, hogy ekkor `T(i,i)` biztosan igaz, `T(i,i+1)` pedig akkor igaz, ha `S(i)=S(i+1)`.

`T(i,j)` értékét pedig számíthatjuk `i` és `j` távolságának növelésével, azaz `T(i,j)` akkor igaz, ha `S(i)=S(j)` és `T(i+1,j-1)` igaz értékű.

Az `R(0)` érték biztosan 0, az `R(1)` érték pedig biztosan 1 lesz. Az `R(i)` értéke biztosan nem lehet több, mint `R(i-1)+1`. Akkor lehet ennél kisebb, ha van olyan `j` index, amelyre `S(j..i)` tükörszó, és az első `j-1` karakterhez tartozó megoldás+1 kisebb az első i karakter minden más lehetséges felbontásához tartozó megoldásnál:

$$R(i) = \min\left(R(i-1)+1, \min_{j=i-1..1, T(j,i)} R(j-1)+1 \right)$$

```
Megoldás(N,S,R):
  Ciklus i=1-től N-1-ig
    T(i,i):=igaz; T(i,i+1):= S(i)=S(i+1)
  Ciklus vége
  T(N,N):=igaz
  Ciklus k=2-től N-1-ig        {T(i,j) számítása}
    Ciklus i=1-től N-k-ig
      j:=i+k; T(i,j):=S(i)=S(j) és T(i+1,j-1)
    Ciklus vége
  Ciklus vége
  R(0):=0; R(1):=1             {inicializálás}
  Ciklus i=2-től N-ig          {R(i) számítása}
    Min:=R(i-1)+1
    Ciklus j=i-1-től 1-ig -1-esével
      Ha T(j,i) és R(j-1)+1<Min akkor Min:=R(j-1)+1
    Ciklus vége
    R(i):=Min
  Ciklus vége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Villamos (20 pont)

A megoldáshoz szükségünk van egy tömbre, amelyben megmondjuk, hogy az `i`-edik elemből a `j`. forgatási állapotban merre lehet lépni. Ezt egy 4-bites értékkel kódolhatjuk:

```
  { 0. bit - lehet-e balra, 1. bit - lehet-e felfelé,
    2. bit - lehet-e jobbra, 3. bit - lehet-e lefelé lépni}
lép: tömb('0'..'4',0..3, byte) =((0,0,0,0),(3,6,12,9),
          (5,10,5,10),(7,14,13,11),(15,15,15,15))
```

Ezek után a kezdőhelyről a célhelyre egy szélességi bejárással keressük meg a legrövidebb utat. Az egyik megoldásban forgatni is lehet, tehát ott a forgatások is lépésnek számítanak!

```
Bejárás(forgat,ksor,kosz,csor,cosz,lruthossz)
  volt(,):=0; SorInicializálás;
  lruthossz:=0; volt(ksor,kosz):=1; táv(ksor,kosz):=0
  Sorba(ksor,kosz)
  Ciklus amíg a SOR nem üres és volt(csor,cosz)=0
    Sorból(sor,osz)
    Felfelélép(sor,osz); Balralép(sor,osz)
    Lefelélép(sor,osz); Jobbralép)sor,osz)
    Forgat(sor,oszlop)
  Ciklus vége
  Ha volt(csor,cosz)>0 akkor lruthossz:=táv(csor,cosz)
                       különben lruthossz:=-1
Eljárás vége.
```

```
Felfelélép(sor,osz):
  Ha sor>1 és volt(sor-1,osz)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz)) és 2)>0 és
       (lép(telem(sor-1,osz),tforg(sor-1,osz)) és 8)>0
      akkor Sorba(sor-1,osz); volt(sor-1,osz):=1
            táv(sor-1,osz):=táv(sor,osz)+1
Eljárás vége.

Balralép(sor,osz):
  Ha osz>1 és volt(sor,osz-1)=0 akkor
     Ha (lép(telem(sor,osz),tforg(sor,osz)) és 1)>0 és
        (lép(telem(sor,osz-1),tforg(sor,osz-1)) és 4)>0
       akkor Sorba(sor,osz-1); volt(sor,osz-1):=1
             táv(sor,osz-1):=táv(sor,osz)+1
Eljárás vége.

Lefelélép(sor,osz):
  Ha sor<n és volt(sor+1,osz)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz)) és 8)>0 és
       (lép(telem(sor+1,osz),tforg(sor+1,osz)) és 2)>0
      akkor Sorba(sor+1,osz); volt(sor+1,osz):=1
            táv(sor+1,osz):=táv(sor,osz)+1
Eljárás vége.

Jobbralép)sor,osz):
  Ha osz<m és volt(sor,osz+1)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz)) és 4)>0 és
       (lép(telem(sor,osz+1),tforg(sor,osz+1)) és 1)>0
      akkor Sorba(sor,osz+1); volt(sor,osz+1):=1
            táv(sor,osz+1):=táv(sor,osz)+1
Eljárás vége.

Forgat(sor,oszlop):
  Ha forgat és volt(sor,osz)<4 és telem(sor,osz)>'0'
       és telem(sor,osz)<'4'
    akkor Sorba(sor,osz); volt(sor,osz):=volt(sor,osz)+1
          táv(sor,osz):=táv(sor,osz)+1
          tforg(sor,osz):=(tforg(sor,osz)+1) mod 4
Ejárás vége.
```

2. feladat: Mozgat (20 pont)

Az első részfeladatban az első 10 sorra hajtsuk végre sorban a `K` műveletet, s számoljuk, hogy mi a sor új pozíciója! A második részfeladatban pedig azt nézzük meg, hogy a végén az első 10 sorba került elemek honnan érkeztek, ezt a műveletek sorrendjében visszafelé kell vizsgálni.

```
Mozgat(K,Tól,Ig,Hova,Poz,RPoz):
  Ciklus i=1-től 10-ig
    p:=i
    Ciklus j=1-től K-ig
      p:=ÚjPoz(p,Tól(j),Ig(j),Hova(j))
    Ciklus vége
    Poz(i):=p
  Ciklus vége
```

```
  Ciklus i=1-től 10-ig
    p:=i
    Ciklus j=K-tól 1-ig -1-esével
      p:=RevPoz(p,Tól(j),Ig(j),Hova(j))
    Ciklus vége
    RPoz(i):=p
  Ciklus vége
Eljárás vége.
```

Ha egy szakasz előrefelé mozog akkor két esetet kell megvizsgálnunk. Az egyikben az előre mozgó szakasz az adott elem mögött volt, az új helye pedig előtte – ilyenkor a pozíció a szakasz hosszával nő. A másikban az adott elem a mozgó szakaszban van – ekkor pedig a pozíció az elmozdulás távolságával csökken.

Ha egy szakasz hátrafelé mozog, akkor is két eset van. Ha az adott elem a mozgó szakaszban van, akkor az elmozdulás távolságával növeljük a helyét. Ha pedig a szakasz átugorja az adott elemet, akkor az a szakasz hosszával előre mozdul.

```
ÚjPoz(Poz,Tól,Ig,Hova):
  hány:=Ig-Tól+1; ÚjPoz:=Poz
  Ha Hova<Tól akkor                  {Hova<Tól≤Ig}
    Ha Hova<Poz és Poz<Tól akkor ÚjPoz:=Poz+hány
    különben ha Tól≤Poz és Poz≤Ig
                akkor ÚjPoz:=Poz+Hova-Tól+1
  különben                        {Tól≤Ig≤Hova}
    Ha Tól≤Poz és Poz≤Ig akkor ÚjPoz:=Poz+Hova-Ig
    különben ha Ig<Poz és Poz≤Hova akkor ÚjPoz:=Poz-hány
Függvény vége.
```

A visszafelé követésnél nem az elmozdulás előtti, hanem az elmozdulás utáni helyet kell nézni, továbbá minden elmozdulást meg kell fordítani.

```
RevPoz(Poz,Tól,Ig,Hova)
  hány:=Ig-Tól+1; RevPoz:=Poz
  Ha Hova<Től akkor                  {Hova<Tol≤Ig}
    Ha Hova+hány<Poz és Poz<Től+hány akkor RevPoz:=Poz-hány
    különben ha Hova+1≤Poz és Poz≤Hova+hány
                akkor RevPoz:=Poz+Től-Hova-1
  különben                        {Től≤Ig≤Hova}
    Ha Tol≤Poz és Poz≤Hova-hány) akkor RevPoz:=Poz+hány
    különben ha Hova-hány<Poz és Poz≤Hova
                akkor RevPoz:=Poz-Hova+Ig
Függvény vége.
```

3. feladat: Ütemezés (15 pont)

A megoldásban minden befejezési időhöz csak a legkésőbbi kezdési időt kell tárolnunk, hiszen a rövidebb munka kevesebb másik elvégzését akadályozza meg. Ezután tároljuk az eddigi utolsó befejezési időt, nézzük végig a lehetséges befejező napokat, s ha az azon a napon befejeződő munka később kezdődik az eddigi utolsónál, akkor vegyük bele a megoldásba (a sorszámát állítsuk -1-szeresére)!

```
Ütemezés(MaxNap,Munka):
  M:=0                {a beosztott munkák száma}
  Foglalt:=0          {eddig már foglalt a mester}
  Ciklus B=1-től MaxNap-ig
    Ha Munka[B].MSorsz≠0 és Foglalt<Munka[B].K
      akkor Foglalt:=B; M:=M+1
            Munka[B].MSorsz:=-Munka[B].MSorsz
  Ciklus vége
Eljárás vége.
```

4. feladat: Tükörszó (20 pont)

Az `N` elemű szöveget az `S` vektorban tároljuk. Jelölje `H(i,j)` az `S(i..j)`-ben levő leghosszabb tükörszó hosszát! Nyilvánvaló, hogy `H(i,i)=1`. `H(i,i+1)` értéke 2, ha `S(i)=S(i+1)` különben szintén 1. A többi `H(i,j)` `i` és `j` távolsága szerint növekvő sorrendben számolható, azaz a feladat megoldható dinamikus programozással:

$$H(i,j)=\max\begin{cases} H(i,j-1) \\ H(i+1,j) \\ H(i+1,j-1)+2 \ \ ha \ \ S(i)=S(j) \end{cases}$$

A megoldás a `H(1,N)` értéke lesz.

```
Tükörszó(S,N,H):
  Ciklus i=1-től N-1-ig
    H[i,i]:=1
    Ha S[i]=S[i+1] akkor H[i,i+1]:=2 különben H[i,i+1]:=1
  Ciklus vége
  H[N,N]:=1
  Ciklus k=2-től N-1-ig
    Ciklus i=1-től N-k-ig
      j:=i+k; Maxi:=H[i,j-1]
      Ha Maxi<H[i+1,j] akkor Maxi:=H[i+1,j]
      Ha S[i]=S[j] és Maxi<H[i+1,j-1]+2
         akkor Maxi:=H[i+1,j-1]+2
      H[i,j]:=Maxi
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

2002. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Betét (17 pont)

A megoldásban külön számoljuk az otthon őrzött pénzt és a bankban levőt.

```
Betét(x,s,p,h):
  bank:=x div 100; otthon:=0
  Ciklus i=0-tól h-ig
    Ki: 'Bankban: ',bank,'00, otthon: ',otthon
    Ha bank<10 akkor otthon:=otthon+bank*s
            különben otthon:=otthon+bank*(s+p)
    bank:=bank+otthon div 100; otthon:=otthon mod 100
  Ciklus vége
Eljárás vége.
```

2. feladat: Szomszéd (28 pont)

Az A részfeladat szerint a legfeljebb `s` távolságra levőket kell kiírni, a tábla széleit nem léphetjük át.

```
SzomszédokA(k,l,s):
  Ciklus i=k-s-től k+s-ig
    Ciklus j=l-s-től l+s-ig
      Ha i≠k vagy j≠l akkor
        Ha i≥1 és i≤n és j≥1 és j≤m akkor Ki: i,j
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A B részfeladatnál további feltétel lesz, hogy a két indexpár különbsége legyen kisebb `s`-nél!

```
SzomszédokB(k,l,s):
  Ciklus i=k-s-től k+s-ig
    Ciklus j=l-s-től l+s-ig
      Ha (i≠k vagy j≠l) és |i-k|+|j-l|≤s akkor
        Ha i≥1 és i≤n és j≥1 és j≤m akkor Ki: i,j
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A C részfeladat az A-tól abban különbözik, hogy a tábla szélein átléphetünk a túlsó szélére.

```
SzomszédokC(k,l,s):
  Ciklus i=k-s-től k+s-ig
    Ciklus j=l-s-től l+s-ig
      Ha i≠k vagy j≠l akkor
        Ha i<1 akkor ii:=i+n
        különben ha i>n akkor ii:=ii-n különben ii:=i
        Ha j<1 akkor jj:=j+m
        különben ha j>m akkor jj:=jj-m különben jj:=j
        Ki: ii,jj
      Elágazás vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A D részfeladat az B-től szintén abban különbözik, hogy a tábla szélein átléphetünk a túlsó szélére.

```
SzomszédokD(k,l,s):
  Ciklus i=k-s-től k+s-ig
    Ciklus j=l-s-től l+s-ig
      Ha (i≠k vagy j≠l) és |i-k|+|j-l|≤s akkor
        Ha i<1 akkor ii:=i+n
        különben ha i>n akkor ii:=ii-n különben ii:=i
        Ha j<1 akkor jj:=j+m
        különben ha j>m akkor jj:=jj-m különben jj:=j
        Ki: ii,jj
      Elágazás vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

3. feladat: DNS (30 pont)

Az S szöveg leghosszabb ismétlődő szakasza legfeljebb az S karakterszámának fele hosszúságú. Mivel a szöveg hossza kicsi, ezért kipróbálhatjuk ettől kezdve visszafelé haladva az összes lehetséges hosszúságot, keresve hogy olyan hosszúságú szakasz ismétlődik-e. Ha a `van` változó hamis marad, akkor nincs ismétlődő szakasz, ha igaz értékű, akkor pedig a `j`-edik karakternél kezdődő `i` hosszúságú rész a legtöbbet ismétlődő.

```
Leghosszabb(S,j,i):
  van:=hamis; i:=hossz(S) div 2
  Ciklus amíg i>2 és nem van
    j:=1
    Ciklus amíg nem van és j+i+i-1≤hossz(S)
      k:=j+i-1
      Ciklus amíg nem van és k+i-1≤ hossz(S)
        k:=k+1; van:=S(j..j+i-1)=S(k..k+i-1)
      Ciklus vége
      Ha nem van akkor j:=j+1
    Ciklus vége
    Ha nem van akkor i:=i-1
  Ciklus vége
Eljárás vége.
```

Megfigyelhetjük, hogy a második feladat megoldásában elég a legtöbbet ismétlődő 3 hosszúságú szakaszt megkeresni. Ha ugyanis van 3-nál hosszabb ismétlődő szakasz, annak az első három betűje is ugyanannyiszor ismétlődik. A `maxdb` 1 marad, ha nincs ismétlődő szakasz, egyébként pedig a `maxj`-edik karakternél kezdődő legtöbbet ismétlődő 3 karakteres szakaszok száma.

```
Ismétlés(S,maxdb,maxj):
  maxdb:=1; maxj:=1
  Ciklus j=1-től hossz(S)-5-ig
    db:=1
    Ciklus k=j+3-tól hossz(S)-2-ig
      Ha S(j..j+2)=S(k..k+2) akkor db:=db+1
    Ciklus vége
    Ha db>maxdb akkor maxdb:=db; maxj:=j
  Ciklus vége
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Épít (20 pont)

A bemenő adatokból építsünk fel egy gráfot, melynek pontjai a munkák, élek pedig azon munkák között legyenek, amelyek között megelőzési reláció ál fenn.

Az elvégzendő munkák sorába az első napon elvégzendők közé biztosan betehetők azok a munkák, amelyeknek nincs előfeltétele, azaz a gráfban nekik megfelelő pont 0 befokú. Ezután nézzük végig ezeket a pontokat, s minden belőlük vezető élt töröljünk a gráfból. Most újra lesznek 0 befokú pontok, ezek lesznek a második napon elvégzendők, … Addig haladunk így tovább a munkákon, amíg mind el nem fogy.

```
Épít(N,BeFok,KiFok,G,Sor):
  i:=0
  Ciklus x=1-től N-ig
    Ha BeFok(x)=0 akkor i:=i+1; Sor(i):=x
  Ciklus vége
  Be(0):=0; Be(1):=i; M:=1; vége:=(Be(1)=0)
  Ciklus amíg nem vége
    Uj:=Be(M)
    Ciklus i=Be(M-1)+1-től Be(M)-ig
      x:=Sor(i)
      Ciklus j=1-től KiFok(x)-ig
        y:=G(x,j); BeFok(y):=BeFok(y)-1
        Ha BeFok(y)=0 akkor Uj:=Uj+1; Sor(Uj):=y
      Ciklus vége
    Ciklus vége
    Ha Be(M)<Uj akkor M:=M+1; Be(M):=Uj különben vége:=igaz
  Ciklus vége
Eljárás vége.
```

2. feladat: Felvételi (20 pont)

Jelöljük `H(i)`-vel az `i`-edik jelentkező adatait! Első lépésként rendezzük őket sorba számlálva szétosztó rendezéssel az elért pontszámuk szerint.

```
Rendezés(H,MaxP,N,RSor):
  Ciklus j=0-tól MaxP+1-ig
    Hány(j):=0
  Ciklus vége
  Ciklus i=1-től N-ig
    Hány(H(i).Pont):=Hány(H(i).Pont)+1
  Ciklus vége
  Ciklus j=MaxP-1-től 0-ig
    Hány(j):=Hány(j)+Hány(j+1)
  Ciklus vége
  Száml:=Hány
  Ciklus i=1-től N-ig
    P:=H(i).Pont; RSor(Száml(P)):=i; Száml(P):=Száml(P)-1
  Ciklus vége
Eljárás vége.
```

A megoldást a pontszám szerint csökkenő sorrenben iterálva kaphatjuk meg.

```
Meg(MaxE,Felvett,PHatar,MaxPont,MaxP,MinP,H,N,RSor,Hany):
  Ciklus e=1-től MaxE-ig
    Felvett(e):=0; PHatár(e):=MinP; MaxPont(e):=MaxP
  Ciklus vége
                      {a P pontszámúak feldolgozása}
  Ciklus P:=MaxP-től MinP-ig -1-esével
  {iterálás a P pontszámúakra, amíg vm. egy. ponthatára>P}
    Ciklus
      Kész:=igaz
      Ciklus e=1-től Esz-ig
        PFelvett(e):=0
      Ciklus vége
      Ciklus ii=Hany(P+1)+1-től Hany(P)-ig
        i:=Rsor(ii); j:=1
        Ciklus          {az i. jelölt melyikre kerülne be?}
          e:=H(i).PSor(j)
          Ha H(i).Pont<PHatár(e) akkor e:=0; j:=j+1
                                 különben j:=MaxPSor+1
        amíg j≤MaxPSor
        Ciklus vége
        Ha e>0 akkor
           Pfelvett(e):=Pfelvett(e)+1
           Ha Felvett(e)+PFelvett(e)>Keret(e) akkor
              {ez már nem fér be, emelni kell a ponthatárt}
              PHatár(e):=P+1
              PFelvett(e):=0;   {nem veszünk fel P pontost}
              Kész:=hamis   {újabb iteráció a P pontosokra}
              Kilép
           Elágazás vége
        Elágazás vége
        Hova(i):=e
      Ciklus vége
    amíg nem Kész
    Ciklus vége
    Ciklus e=1-től Esz-ig
      Felvett(e):=Felvett(e)+PFelvett(e)
      Ha PFelvett(e)>0 és MaxPont(e)>P akkor MaxPont(e):=P
    Ciklus vége
  Ciklus vége
  Ciklus e=1-től Esz-ig
    Ha Felvett(e)=0 akkor MaxPont(e):=MinP
  Ciklus vége
Eljárás vége.
```

3. feladat: Hatszög (18 pont)

A feladat ábrája szerint a hatszög alapú tér mezőit három index azonosítja, a szövegből azonban egyértelműen következik, hogy egy tetszőleges (`i,j,k`) hely szomszédjai indexe olyan, hogy az egyik eggyel csökken, a másik eggyel nő, a harmadik pedig változatlan marad, azaz a harmadik index az első kettőből számítható. Ebből belátható, hogy mivel a (0,0,0) pont van középen, `k=-i-j`.

A feladat ezek után egy egyszerű szélességi gráfbejárás, amelyben az elért pontok magasságát -1-szeresére változtatjuk.

```
Bejár(p,q,r,x,y,z,lr):
  Felvesz(p,q,0)
  Ciklus amíg a SOR nem üres és t(x,y)>0
    Sorból(p,q,lr); lr:=lr+1; r:=-p-q
    Ha t(p-1,q+1)=-t(p,q) akkor Felvesz(p-1,q+1,lr)
    Ha t(p-1,q)=-t(p,q) akkor Felvesz(p-1,q lr)
    Ha t(p+1,q-1)=-t(p,q) akkor Felvesz(p+1,q-1,lr)
    Ha t(p+1,q)=-t(p,q) akkor Felvesz(p+1,q,lr)
    Ha t(p,q-1)=-t(p,q) akkor Felvesz(p,q-1,lr)
    Ha t(p,q+1)=-t(p,q) akkor Felvesz(p,q+1,lr)
  Ciklus vége
  Ha t(x,y)>0 akkor lr:=-1
Eljárás vége.

Felvesz(p,q,lr):
  Sorba(p,q,lr), t(p,q):=-t(p,q)
Eljárás vége.
```

4. feladat: Lefed (17 pont)

Vegyük észre, hogy ha az (x,y) pont le van fedve, akkor az (y,x) pont is, ezért ha a. bemenetben x>y, akkor az (y,x) pontot vesszük fel. Ha van (x,x) koordinátájú lefedendő pont akkor az (x,x)-en átmenő egyenespárt be kell választani. Tároljuk a lefedendő pontokat a T logikai tömbben.

A feladat visszalépéses kereséssel oldható meg. Egy megoldás megadható egy A bitvektorral, A(x) akkor és csak akkor igaz, ha az (x,x) ponton átmenő egyenespárt szerepel a megoldásban. Az A vektort lépésenként építi a Keres rekurzív eljárás. A keresés nagymértékben gyorsítható, ha figyelembe vesszük, hogy ha x-et nem választjuk, akkor minden olyan y-t választani kell, amely T(x,y)=igaz és y-t még nem választottuk A választás visszaállítása egyszerűbb lesz, ha az A vektor nem logikai vektor, hanem A(x):=y, ha a Keres eljárásban x-et nem választjuk, de y-t választani kell.

```
Számít(N,T):
  M:=0; OptM:=N+1
  Ciklus x=1-től N-ig
    Ha T(x,x) akkor A[x]:=x; M:=M+1 különben A(x):=0
  Ciklus vége
  Keres(1)
  Ki: OptM
  Ciklus x=1-től N-ig
   Ha OptA(x)>0 akkor Ki: x
  Ciklus vége
Eljárás vége

Keres(x):
  Ha x>N akkor
    Ha M<OptM akkor                        {jobb megoldást találtunk}
      OptA:=A; OptM:=M                     {feljegyezzük}
  különben ha A(x)>0 akkor Keres(x+1)      {x-et már beválasztottuk}
  különben
    h:=0                                   {először x-et nem választjuk be}
    Ciklus y=x+1-től N-ig
      Ha A(y)=0 és T(x,y) akkor            {y-t be kell választani}
        A(y):=x; M:=M+1; h:=h+1            {jegyezzük a beválasztást}
    Ciklus vége
    Ha h=0 akkor Keres(x+1)          {nincs lefedendő pont az x-oszlopban}
```

```
    különben ha M<OptM akkor Keres(x+1)
             Ciklus y:=x+1-től N-ig          {visszaállítás}
               Ha A(y)=x akkor A(y):=0; M:=M+1
             Ciklus vége
             A[x]:=x; M:=M+1                 {x-et válaszjuk, bejegyzés}
             Keres(x+1)
             A(x):=0; M:=M-1                 {visszaállítás}
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Építkezés (16 pont)

A bemenő adatokból építsünk fel egy gráfot, melynek pontjai a munkák, élek pedig azon munkák között legyenek, amelyek között megelőzési reláció ál fenn! A feladat szerint ugyan kétféle ilyen kapcsolat is van (speciális munka, illetve megelőzés), de a gráf felépítése szempontjából ezek egyformák.

Az elvégzendő munkák sorába az első napon elvégzendők közé biztosan betehetők azok a munkák, amelyeknek nincs előfeltétele, azaz a gráfban nekik megfelelő pont 0 befokú. Ezután nézzük végig ezeket a pontokat, s minden belőlük vezető élt töröljünk a gráfból. Most újra lesznek 0 befokú pontok, ezek lesznek a második napon elvégzendők, … Addig haladunk így tovább a munkákon, amíg mind el nem fogy.

```
Építkezés(N,BeFok,KiFok,G,Sor):
  i:=0
  Ciklus x=1-től N-ig
    Ha BeFok(x)=0 akkor i:=i+1; Sor(i):=x
  Ciklus vége
  Be(0):=0; Be(1):=i; M:=1; vége:=(Be(1)=0)
  Ciklus amíg nem vége
    Uj:=Be(M)
    Ciklus i=Be(M-1)+1-től Be(M)-ig
      x:=Sor(i)
      Ciklus j=1-től KiFok(x)-ig
        y:=G(x,j); BeFok(y):=BeFok(y)-1
        Ha BeFok(y)=0 akkor Uj:=Uj+1; Sor(Uj):=y
      Ciklus vége
    Ciklus vége
    Ha Be(M)<Uj akkor M:=M+1; Be(M):=Uj különben vége:=igaz
  Ciklus vége
Eljárás vége.
```

2. feladat: Kép (20 pont)

Első lépésként határozzuk meg a keresett kis kép kódolt változatát! (A feladat kitűzésének évében a nagy kép kódolatlan változata nem fért el a memóriában, így a megoldásban mindenképpen két kódolt képpel kellett dolgozni.) Legyen `keres(i).p(j)` az `i`-edik sor `j`-edik kódja: egy 255-nél nem nagyobb darabszám és egy színkód! Ez beolvasáskor előállítható.

```
keres:tömb(1..maxk,
           rekord(darab: byte,
                  p: tömb(1..maxk,rekord(db,szin: egész))))
```

```
Előfeldolgozás(keres,k):
  Ciklus i=1-től k-ig
    e:=-1; j:=0
    Ciklus amíg nem sorvége?(f)
      Olvas(f,d)
      Ha d=e és keres(i).p(j).db<255
         akkor keres(i).p(j).db:=keres(i).p(j).db+1
      különben j:=j+1; keres(i).p(j).szin:=d
               keres(i).p(j).db:=1
      e:=d
    Ciklus vége
    keres(i).darab:=j
  Ciklus vége
Eljárás vége.
```

A nagyobb képből elég egyszerre K sornak a memóriában lenni, majd ezekben már kereshetünk. Ha nem találtuk meg a kis képet az adott K sorban, akkor olvassuk a következő sort.

```
Keresés(n,van,x,y):
  van:=hamis
  Ciklus i=1-től k-1-ig
    Olvas(i-1,sor((i-1) mod k)
  Ciklus vége
  i:=1
  Ciklus amíg nem van és i≤n-k+1
    Olvas(i+k-2,sor((i+k-2) mod k)
    Ha Jósor(i,j) akkor van:=igaz; x:=i; y:=j
               különben i:=i+1
  Ciklus vége
Eljárás vége.
```

A `Jósor(i,j)` függvény igaz értékű lesz, ha az `i`-edik sorban kezdve a `j`-edik pozíciótól található meg a kis kép. Ehhez az `i`-edik sor összes lehetséges kezdőpozícióját meg kell vizsgálnunk.

```
Jósor(i,j):
  j:=1
  Ciklus amíg j≤n-k+1 és nem Jópozíció(i,j)
    j:=j+1
  Ciklus vége
  Jósor:=(j≤n-k+1)
Függvény vége.
```

Egy pozíció akkor jó, ha ettől kezdődően `k` darab sorban `k` pozíción megegyezést találunk.

```
Jópozíció(i,j):
  q:=(i-1) mod k; db:=1
  Ciklus amíg db≤k és nem rosszsor(q,db,j)
    q:=(q+1) mod k; db:=db+1
  Ciklus vége
  Jópozíció:=(db>k)
Függvény vége.
```

Egy soron belül úgy nézhetjük meg a k darab pozíciót, hogy először megkeressük az első pozíciót, ami egy kód közepén is lehet, tehát ki kell fejtenünk. Utána lépkedünk a kódokon, egyenlőséget vizsgálva, majd a legvégén újra lehetséges, hogy bontanunk kell egy kódot.

```
Rosszsor(q,db,j):
  t:=1; r:=sor(q).p(t).db
  Ciklus amíg r<j
    t:=t+1; r:=r+sor(q).p(t).db
  Ciklus vége
  r:=r-j+1; s:=1
  Ciklus amíg s≤keres(db).darab és r=keres(db).p(s).db és
              sor(q).p(t).szin=keres(db).p(s).szin
    s:=s+1; t:=t+1; r:=sor(q).p(t).db
  Ciklus vége
  Rosszsor:=s<keres(db).darab vagy s<keres(db).darab és
            nem(sor(q).p(t).szin=keres(db).p(s).szin és
                r<keres(db).p(s).db)
Függvény vége.
```

3. feladat: Robot (19 pont)

Az A részfeladatban minden tárgyat össze kell gyűjteni. Mivel balra nem léphetünk, ezért az első oszlopban lefelé kell menni a legalsó tárgyig. Ha ez a sor fölött a következő oszlopban nincs tárgy, akkor jobbra léphetünk, majd lefelé indulhatunk. Ha ez a sor alatt nincs a következő oszlopban tárgy, akkor szintén léphetünk jobbra, majd felfelé indulhatunk. Ha azonban alatta és fölötte is van tárgy, akkor még nem mehetünk jobbra, mert az egyik irányban levőket nem érhetjük el. Ez azt jelenti, hogy a haladási irányunkban (az első oszlopban ez lefelé irány volt) tovább kell menni addig, amíg el nem érjük a következő oszlop legutolsó (azaz lefelé haladás esetén legalsó) elemét.

A megoldáshoz tehát a beolvasáskor elég azt tárolni,hogy mi az a legkisebb x, amelyre az (x,y) mezőn van tárgy (MinX(y)), valamint a legnagyobb x, amelyre az (x,y) mezőn van tárgy (MaxX(y)).

Jelölje EFel, illetve ELe az előző oszlopig az optimális megoldást, felfelé és lefelé haladva! Fel és Le pedig legyen az optimális megoldás az aktuális oszlopig, felfelé és lefelé haladva!

A feladat szerint az első oszlop 1-es pozíciójától indulunk, azaz MinX(1)=1, és az N-edik oszlop N-edik pozíciójára kell érkeznünk, azaz MaxX(N)=N. Ha az első vagy az utolsó oszlopban nincs tárgy, akkor ott a MinX értéke egyenlő a MaxX értékével.

```
Robot:
  MinX(1):=1; Ha MaxX(1)=0 akkor MaxX(1):=1
  MaxX(N):=N; Ha MinX(N)>N akkor MinX(N):=N;
  EFel:=0; ELe:=0; y0:=0; Minx(0):=1; MaxX(0):=1
  Ciklus y=1-től N-ig
    Ha MaxX(y)>0 akkor     {van tárgy az y. oszlopban}
                                {Lefele menet}
       Le:=EFel+(y-y0)+|MinX(y)-MinX(y0)|+MaxX(y)-MinX(y)
       Tól(Lefele,y):=Felfele; Lép:=Inf
       Ha MaxX(y0)≤MinX(y) vagy y0+1<y akkor
          Lép:=ELe+(y-y0)+|MaxX(y0)-MinX(y)|+MaxX(y)-MinX(y)
       Ha Lép<Le akkor Le:=Lép; Tól(Lefele,y):=Lefele
```

```
                                    {Felfele menet}
        Fel:=ELe+(y-y0)+|MaxX(y)-MaxX(y0)|+MaxX(y)-MinX(y)
        Tól(Felfele,y):=Lefele; Lép:=Inf
        Ha MinX(y0)≥MaxX(y) vagy y0+1<y akkor
           Lép:=EFel+(y-y0)+|MinX(y0)-MaxX(y)|+MaxX(y)-MinX(y)
        Ha Lép<Fel akkor Fel:=Lép; Tól(Felfele,y):=Felfele
        y0:=y; EFel:=Fel; ELe:=Le
      Elágazás vége
    Ciklus vége
    Ha Fel<Le akkor Megoldás:=Fel-1; Irány:=Felfele
           különben Megoldás:=Le-1; Irány:=Lefele
    y:=N
    Ciklus amíg y>0
      Hol(y):=Irány; Irány:=Tól(Irany,y)
      Ciklus
        y:=y-1
      amíg MaxX(y)=0
      Ciklus vége
    Ciklus vége
Eljárás vége.
```

4. feladat: Villamos (20 pont)

A megoldáshoz szükségünk van egy tömbre, amelyben megmondjuk, hogy az `i`-edik elemből a `j`. forgatási állapotban a `k`. irányból belépve merre lehet lépni. Ezt egy 4-bites értékkel kódolhatjuk:

```
  { 0. bit - lehet-e balra, 1. bit - lehet-e felfelé,
    2. bit - lehet-e jobbra, 3. bit - lehet-e lefelé lépni}
lép: tömb('0'..'4',0..3,0..3, byte)
       =(((0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0)),
         ((9,0,0,9),(3,3,0,0),(0,6,6,0),(0,0,12,12)),
         ((5,0,5,0),(0,10,0,10),(5,0,5,0),(0,10,0,10)),
         ((13,0,5,9),(3,11,0,10),(5,6,7,0),(0,10,12,14)),
         ((13,6,7,9),(3,11,12,14),(13,6,7,9),(3,11,12,14)),
         ((11,3,0,9),(3,7,6,0),(0,6,14,12),(9,0,12,13)),
         ((9,6,6,9),(3,3,12,12),(9,6,6,9),(3,3,12,12)),
         ((15,3,5,9),(3,15,6,10),(5,6,15,12),(9,10,12,15)))
```

Ezek után a kezdőhelyről a célhelyre egy szélességi bejárással keressük meg a legrövidebb utat. Az egyik megoldásban forgatni is lehet, tehát ott a forgatások is lépésnek számítanak!

```
Bejárás(forgat,ksor,kosz,kir,csor,cosz,lruthossz)
  volt(,):=0; SorInicializálás;
  lruthossz:=0; volt(ksor,kosz):=1; táv(ksor,kosz):=0
  Sorba(ksor,kosz,kir)
  Ciklus amíg a SOR nem üres és volt(csor,cosz)=0
    Sorból(sor,osz,ir)
    Felfelélép(sor,osz); Balralép(sor,osz)
    Lefelélép(sor,osz); Jobbralép(sor,osz)
  Ciklus vége
  Ha volt(csor,cosz)>0 akkor lruthossz:=táv(csor,cosz)
                       különben lruthossz:=-1
Eljárás vége.
```

```
Felfelélép(sor,osz):
  Ha sor>1 és (volt(sor-1,osz) és 8)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz),ir) és 2)>0
      akkor Sorba(sor-1,osz,3)
            volt(sor-1,osz):=volt(sor-1,osz) vagy 8
            táv(sor-1,osz):=táv(sor,osz)+1
Eljárás vége.

Balralép(sor,osz):
  Ha osz>1 és (volt(sor,osz-1) és 4)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz),ir) és 1)>0
      akkor Sorba(sor,osz-1.2)
            volt(sor,osz-1):=volt(sor,osz-1) vagy 4
            táv(sor,osz-1):=táv(sor,osz)+1
Eljárás vége.

Lefelélép(sor,osz):
  Ha sor<n és (volt(sor+1,osz) és 2)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz,ir)) és 8)>0
      akkor Sorba(sor+1,osz,1)
            volt(sor+1,osz):=volt(sor+1,osz) vagy 2
            táv(sor+1,osz):=táv(sor,osz)+1
Eljárás vége.

Jobbralép(sor,osz):
  Ha osz<m és (volt(sor,osz+1) és 1)=0 akkor
    Ha (lép(telem(sor,osz),tforg(sor,osz),ir) és 4)>0
      akkor Sorba(sor,osz+1,0)
            volt(sor,osz+1):=volt(sor,osz+1) vagy 1
            táv(sor,osz+1):=táv(sor,osz)+1
Eljárás vége.
```

Egy lehetséges másik megoldásban `Pe[i,ir]` legyen azon irányok halmaza, amelyre az `i`. pályaelemre az `ir` irányból belépve ki lehet lépni!

```
Pe: Tömb[0..7,0..3,Halmaz(0..3)))
    =(({},{},{},{}),
      ({0,3},{},{},{0,3}),
      ({0,2},{},{0,2},{}),
      ({0,2,3},{},{0,2},{0,3}),
      ({0,2,3},{1,2},{0,1,2},{0,3}),
      ({0,1,3},{0,1},{},{0,3}),
      ({0,3},{1,2},{1,2},{0,3}),
      ({0,1,2,3},{0,1},{0,2},{0,3}))
```

A `Keres` eljárás a `Tól` pozíciótól az `Ig` pozícióig vezető legrövidebb utat számítja, szélességi bejárással. Csak akkor hívjuk meg, ha kezdetben nem a célpontban vagyunk, ilyenkor ugyanis a megoldás a 0 lépésszám. A `NemVolt(x,y,ir)` jelentse, hogy itt még nem jártunk! A táblát vegyük körbe mindenhol hamis értékekkel, nehogy kimenjünk a vizsgált részből!

```
Keres(Tólx,Tóly,HolBe,Igx,Igy):
  P.x:=Tólx; P.y:=Tóly; P.ir:=HolBe; P.Lep:=0; Keres:=-1
  NemVolt(P.x,P.y,HolBe):=hamis           {itt már jártunk}
  Sorinicializálás(S); Sorba(S,P); kész:=hamis
  Ciklus amíg a SOR nem üres és nem kész
    Sorból(S,P); Q.Lép:=P.Lép+1
    Kilép(T(P.x,P.y).Pes,T(P.x,P.y).f,P.ir,KiSz)
                      {KiSz-beli irányokban léphetek innen}
    Ciklus i=0-tól 3-ig
      Ha i∈KiSz akkor              {lépés az i. irányában}
         Q.ir:=Be(i)                     {a belépés iránya}
         Q.x:=P.x+Dx(i); Q.y:=P.y+Dy(i)
                                  {a szomszéd koordinátái}
         Ha NemVolt(Q.x,Q.y,Q.ir) és     {lehet P->Q lépés}
              BeLép(T(Q.x,Q.y).Pes,T(Q.x,Q.y).f,Q.ir) akkor
           Ha Igx=Q.x és Igy=Q.y akkor Keres:=Q.Lep
                                       kész:=igaz
           NemVolt(Q.x,Q.y,Q.ir):=hamis; SorBa(S,Q)
                        {bejegyezzük, hogy itt már jártunk}
      Elágazások vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Legyen `Sz` azon irányok halmaza, amely irányokba ki lehet lépni, ha az `ir` irányból lépünk be a `Pes` sorszámú, `f` forgatású pályaelemre!

```
KiLép(Pes,f,ir,Sz):
  i:=(4-f+ir) Mod 4; Sz:=()
  Ciklus j=0-tól 3-ig
    Ha j∈Pe(Pes,i) akkor Sz:=Sz∪{(j+f) mod 4)}
  Ciklus vége
Eljárás vége.
```

Itt azt vizsgáljuk, hogy beléphet-e a `Pes` sorszámú, `f`-el elforgatott pályaelemre az `ir` irányból?

```
BeLép(Pes,f,ir):
  BeLép:=Pe(Pes,(4-f+ir) Mod 4)≠{}
Függvény vége.
```

2003. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendezés (25 pont)

A. 1 lépés 5 pont

B. 2 lépés 5 pont

C. 7 lépés 5 pont

D. 5 lépés 5 pont

E. 9 lépés 5 pont

2. feladat: Keresés (20 pont)

A1. Alfa,Béta,Gamma,Delta igen (helyes válaszonként 1 pont) max. 4 pont

A2. Alfa, Béta nem; Gamma,Delta igen (helyes válaszonként 1 pont) max. 4 pont

A3. Alfa, Béta, Gamma, Delta nem (helyes válaszonként 1 pont) max. 4 pont

A4. Alfa nem; Béta,Gamma,Delta igen (helyes válaszonként 1 pont) max. 4 pont

A5. Alfa, Béta,Gamma,Delta nem (helyes válaszonként 1 pont) max. 4 pont

3. feladat: Szűrés (23 pont)

Első: kihagyja az összes szóközt 6 pont

Második: egymás melletti szóközökből pontosan egyet hagy meg 6 pont

Harmadik: egymás melletti szóközökből pontosan egyet hagy meg, 6 pont
de ha a sor elején szóközök vannak, azokat mind elhagyja 5 pont

Megjegyzés: akkor is jár pont, ha a kihagyottak helyett a megmaradtakat adják meg.

4. feladat: Robot (32 pont)

A.

		1	2	3
		0	11	4
			10	5
			9	6
			8	7

4 pont

B.

15	16	1	2	3
14	17	0	23	4
13	18		22	5
12	19	20	21	6
11	10	9	8	7

8 pont

C.

13	14	1	2	
12	11	0	3	4
	10	9	6	5
		8	7	

5 pont

D.

9	10	11	12	13
8		0	21	14
7		1	20	15
6		2	19	16
5	4	3	18	17

7 pont

E.

3	2	1	18	17
4	5	0	19	16
7	6	13	14	15
8	11	12		
9	10			

8 pont

Ha bármelyik szabály esetén nem teljesen jó a megoldás, akkor a legutolsó helyesen kitöltött sorszám harmadának egészrésze adható. (Például, ha az A szabálynál jó helyen van az 1,2,3,4, 5 sorszám, de a 6 már nem, akkor 5/3 egészrésze, azaz 1 pont adható. A pontszámot ilyenkor nem befolyásolja, hogy a többi sorszám a helyén van-e vagy sem.)

Kilencedik-tizedik osztályosok

1. feladat: Többségi csoport (20 pont)

1. A. Nem 1 pont

2. A. Igen 1 pont
   B. Jó válasz: 5 2 pont

3. A. Igen 1 pont
   B. Jó válasz: 2, 5 2 pont

4. A. Nem 1 pont

5. A. Igen 1 pont

   B. Jó válasz: 1, 2, 3 2 pont

6. A. Igen 1 pont
   B. Jó válasz: 1, 2, 3, 4 2 pont

7. A. Igen 1 pont

   B. Jó válasz: 11 2 pont

8. A. Igen 1 pont
   B. Jó válasz: 6, 7, 8 2 pont

2. feladat: REAL nyelv (19 pont)

A. Hibásak: A2, A3, A6 1+1+1 pont

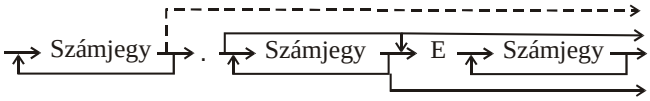
   Helyesek: A1, A4, A5, A7 1+1+1+1 pont

B2. 

2 pont

B3. 

2 pont

B6. 

2 pont

C1. 1.1.1, azaz több pont lehet benne 2 pont

C2. 1.E1E1, azaz több E lehet benne 2 pont

   1.E, azaz az E után lehet, hogy nincs számjegy 2 pont

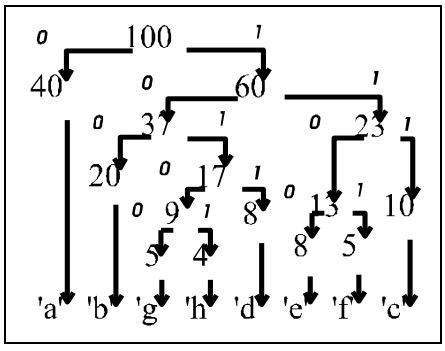
3. feladat: Szűrés (23 pont)

Első: kihagyja az összes szóközt 7 pont

Második: egymás melletti szóközökből pontosan egyet hagy meg 6 pont

Harmadik: egymás melletti szóközökből pontosan egyet hagy meg, 5 pont
   de ha a sor elején szóközök vannak, azokat mind elhagyja 5 pont

4. feladat: Huffman kód (16 pont)



'a' → 0 2 pont

'b' → 100 2 pont

'c' → 111 2 pont

'd' → 1011 2 pont

'e' → 1100 2 pont

'f' → 1101 2 pont

'g' → 10100 2 pont

'h' → 10101 2 pont

5. feladat: FURA nyelv (20 pont)

A. AC szava a FURA nyelvnek 1 pont

ABB⇒ABC⇒ABCBC⇒ABBC⇒AB⇒AC 3 pont

ABBCC szava a FURA nyelvnek 1 pont

ABB ⇒ ABC ⇒ ABBCC 3 pont

ABCCC szava a FURA nyelvnek 1 pont

ABB ⇒ ABC ⇒ ABCBC ⇒ ABCBCBCBC ⇒ ABCBCBC ⇒ ABCBBC
⇒ ABCB ⇒ ABCC ⇒ ABCCBCC ⇒ ABCCC 3 pont

ABBBBBB szava a FURA nyelvnek 1 pont

ABB ⇒ ABBBB ⇒ ABBBBBBBB ⇒ ABBBBBBBC ⇒ABBBBBB 3 pont

B. Minden szó A betűvel kezdődik 2 pont

Csak az első betű lehet A 2 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rendezés (20 pont)

A. 4 lépés 4 pont

B. 5 lépés 4 pont

C. 4 lépés 4 pont

D. 2 lépés 4 pont

E. 3 lépés 4 pont

2. feladat: Halmazok (20 pont)

A. Figyelem: nem szükséges pontosan az alábbi, matematikai jellegű definíciókat megadni!

Első: a két sorozatként ábrázolt halmaz uniója (A∪B) 1 pont
A=(1,3,5), B=(2,3,4) ⇒ C=(1,2,3,4,5) 1 pont

Második: a két sorozatként ábrázolt halmaz metszete (A∩B) 1 pont
A=(1,3,5), B=(2,3,4) ⇒ C=(3) 1 pont

Harmadik: az első halmazból kivonva a második (A–B) 1 pont
A=(1,3,5), B=(2,3,4) ⇒ C=(1,5) 1 pont

Negyedik: a két halmaz szimmetrikus differenciája (A–B∪B–A) 1 pont
A=(1,3,5), B=(2,3,4) ⇒ C=(1,2,4,5) 1 pont

B. Figyelem: itt is lehetnek alternatív megfogalmazások!

Első: ez is unió, de a többször előforduló elemek az előfordulás-számuk maximumával kerülnek az eredménybe 2 pont
A=(1,3,3,3,5), B=(2,3,3,4) ⇒ C=(1,2,3,3,3,4,5) 1 pont

Második: ez is metszet, de a többször előforduló elemek az előfordulás-számuk minimumával kerülnek az eredménybe 2 pont
A=(1,3,3,3,5), B=(2,3,3,4) ⇒ C=(3,3) 1 pont

Harmadik: ez is különbség, de a többször előforduló elemek az A-beli előfordulás-számuk – a B-beli előfordulás-számukszor kerülnek az eredménybe 2 pont
A=(1,3,3,3,5), B=(2,3,3,4) ⇒ C=(1,3,5) 1 pont

Negyedik: ez is szimmetrikus differencia, de a többször előforduló elemek az A-beli előfordulás-számuk – a B-beli előfordulás-számukszor, vagy a B-beli előfordulás-számuk – az A-beli előfordulás-számukszor kerülnek az eredménybe 2 pont
A=(1,3,3,3,5), B=(2,3,3,4) ⇒ C=(1,2,3,4,5) 1 pont

3. feladat: Szűrés (20 pont)

Első: Kihagy minden (és) zárójelet. 2 pont

A (kezdőzárójeltől a következő első) végzárójelig mindent kihagy. 3 pont

Ha a (kezdőzárójelet nem követi) végzárójel, akkor a sor végéig mindent kihagy. 1 pont

Második: Kihagy minden (, [, { és),], } zárójelet. 2 pont

A (, [, { kezdőzárójeltől a következő első valamely),], } végzárójelig (nem kell párnak lenni) mindent kihagy. 3 pont

Ha (, [, { kezdőzárójelet nem követi végzárójel, akkor a sor végéig mindent kihagy. 1 pont

Példa: xx(aa[bb]yy)zz ⇒ xxyyzz

Példa: xx(aa[bb)yy]zz ⇒ xxyyzz

Harmadik: Kihagy minden (, [, { zárójelet. 2 pont

A (, [vagy { kezdő zárójel után a megfelelő végzárójelig nem vesz figyelembe semmilyen kezdő zárójelet. 2 pont

A (, [, { kezdőzárójeltől a következő első megfelelő végzárójelig mindent kihagy. 3 pont

Ha (, [, { kezdőzárójelet nem követi végzárójel, akkor a sor végéig mindent kihagy. 1 pont

Példa: xx(aa[bb)yy]zz ⇒ xxyy]zz

4. feladat: Áramkör (20 pont)

A1. $U_0=0$, $U_1=0$, $U_2=0$ 1+1+1 pont

A2. $U_0=0$, $U_1=1$, $U_2=1$ 1+1+1 pont

A3. $U_0=1$, $U_1=1$, $U_2=0$ 1+1+1 pont

A4. $U_0=0$, $U_1=0$, $U_2=1$ 1+1+1 pont

B. Az áramkör kétbites összeadó, azaz 8 pont

$U_0=(Y_0+Z_0) \bmod 2$, $AT_0=(Y_0+Z_0) \operatorname{div} 2$

$U_1=(Y_1+Z_1+AT_0) \bmod 2$, $AT_1=(Y_1+Z_1+AT_0) \operatorname{div} 2$

$U_2=AT_1$

5. feladat: Nyelv (20 pont)

A. AC szava a FURA nyelvnek 1 pont

ABB⇒ABC⇒ABCBC⇒ABBC⇒AB⇒AC 3 pont

ABBCC szava a FURA nyelvnek 1 pont

ABB ⇒ ABC ⇒ ABBCC 3 pont

ABCCC szava a FURA nyelvnek 1 pont

ABB ⇒ ABC ⇒ ABCBC ⇒ ABCBCBCBC ⇒ ABCBCBC ⇒ ABCBBC ⇒ ABCB ⇒ ABCC ⇒ ABCCBCC ⇒ ABCCC 3 pont

ABB szava a FURA nyelvnek 1 pont

ABB ⇒ ABBBB ⇒ ABBBBBBBB ⇒ ABBBBBBBC ⇒ABBBBBB 3 pont

B. Minden szó A betűvel kezdődik 2 pont

Csak az első betű lehet A 2 pont

2003. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Hangok száma (20 pont)

Vegyünk fel egy tömböt a két- és háromjegyű mássalhangzók számára!

```
h: tömb(1..9,szöveg)
    =('cs','dz','gy','ly','ny','sz','ty','zs','dzs');
```

Haladjunk végig az s szöveg hangjain, ha egymás után nem két azonos mássalhangzót találunk, akkor számoljunk! Ugyancsak számolni kell, ha többjegyű mássalhangzó után vagyunk (pl. sz után z következik).

```
Hangok száma(a):
  db:=0; i:=1; p:=igaz
  Ciklus amíg i≤hossz(s)-2
    Ha p vagy s(i)≠s(i-1) akkor db:=db+1
    i:=i+hossza(i,p)
  Ciklus vége
Eljárás vége.
```

Az i-edik karakteren kezdődő hang hosszát a h tömb alapján számoljuk.

```
hossza(i,p):
  p:=igaz
  Ha s(i)+s(i+1)+s(i+2)=h(9) akkor hossza:=3
  különben k:=1
           Ciklus amíg k≤8 és s(i)+s(i+1)≠h(k)
             k:=k+1
           Ciklus vége
           Ha k≤8 akkor hossza:=2
               különben hossza:=1; p:=hamis
Függvény vége.
```

2. feladat: Eszperantó számok (27 pont)

Az eszperantó számok nevét nagyon egyszerűen képezik, ismerni kell hozzá a számjegyek nevét (jegy tömb), valamint az egyes helyiértékek (tíz, száz, ezer) nevét (helyi tömb).

```
jegy: tömb(1..9,szöveg=('unu','du','tri','kvar','kvin',
                        'ses','sep','ok','nau')
helyi: tömb(1..3,szöveg =('dek','cent','mil')
```

Az ezresek, százasok, tízesek elé nem kell írni semmit, ha az aktuális számjegy 1-es, és egyáltalán nem kell írni semmit, ha 0!

```
Számírás(szám):
  Ha szám≥2000 akkor Ki: jegy(szám div 1000)
  Ha szám≥1000 akkor Ki: helyi(3),' '
  szám:=szám mod 1000
  Ha szám≥200 akkor Ki: jegy(szám div 100)
  Ha szám≥100 akkor Ki: helyi(2),' '
  szám:=szám mod 100
  Ha szám≥20 akkor Ki: jegy(szám div 10)
  Ha szám≥10 akkor Ki: helyi(1),' '
  szám:=szám mod 10
  Ha szám≥1 akkor Ki: jegy(szám)
Eljárás vége.
```

3. feladat: Virág (28 pont)

Tároljuk egy konstans tömbben a növények lehetséges állapotát!

```
kód: tömb(1..5,karakter)=('K','N','V','T','E')
```

Ki kell számolnunk, hogy az induló héten melyik állapotból hány van, majd meg kell határozni, hogy melyik állapotból van a legtöbb! Ezután csak azt kell megnéznünk, hogy ebből mikorra lesz legelőször virágzó állapot.

```
Virágok(kód,sor,oszlop):
  db():=0
  Ciklus i=1-től sor-ig
    Ciklus j=1-től oszlop-ig
      Be: c; k:=1
      Ciklus amíg c≠kód(k)
        k:=k+1
      Ciklus vége
      db(k):=db(k)+1
    Ciklus vége
  Ciklus vége
  k:=1
  Ciklus i=2-től 5-ig
    Ha db(i)>db(k) akkor k:=i
  Ciklus vége
  Ki: 'Hét sorszám: '
  Ha k≤3 akkor Ki: 4-k különben Ki: 9-k
  Ki: 'Virágok száma: ',db(k)
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Mássalhangzók (12 pont)

Egy számlálót (db) kell növelgetnünk minden mássalhangzónál! Ha magánhangzóhoz érünk, akkor a számláló értékét ki kell írni, majd lenullázhatjuk!

```
Mássalhangzók(s,db):
  db:=0
  Ciklus i=1-től hossz(s)-ig
    Ha Magánhangzó(s(i)) akkor Ha db>0 akkor Ír(g,db,' ')
                               db:=0
                       különben db:=db+1
  Ciklus vége
  Ha db>0 akkor Ír(g,db)
Eljárás vége.
```

Egy betű akkor magánhangzó, ha szerepel a magánhangzók halmazában.

```
Magánhangzó(c):
  Magánhangzó:=c∈{'a','e','i','o','u'}
Függvény vége.
```

2. feladat: Képkódolás (18 pont)

Olvassuk a kódokat (kód) a bemeneti állományból! Ha kódot olvasunk, akkor számoljuk, hogy mekkora az aktuális rész mérete (kn), valamint hogy hol van a bal felső sarka (ks, ko)! Ha elérünk a kód végére, akkor az azt követő betűvel töltsük ki a kép (kép) megfelelő részét!

```
Dekódol(n,m,kép):
  Ciklus k=1-től m-ig
    Olvas(f,kód); kl:=1; ks:=1; ko:=1; kn:=n
    Ciklus amíg kód(kl)≠'0'
      kn:=kn div 2
      elágazás
        kód(kl)='1' esetén {nem változik a sarok}
        kód(kl)='2' esetén ko:=ko+kn
        kód(kl)='3' esetén ks:=ks+kn
        kód(kl)='4' esetén ks:=ks+kn; ko:=ko+kn
      Elágazás vége
      kl:=kl+1
    Ciklus vége
    Ciklus i=ks-től ks+kn-1-ig
      Ciklus j=ko-tól ko+kn-1-ig
        kép(i,j):=kód(kl+1)
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

3. feladat: Harmadolás (15 pont)

Minden megbízásnál tároljuk, hogy az adott megbízott elosztotta-e a pénzét háromfelé (osztott), valamint azt, hogy ő és a megbízottjai hányadszor harmadolták az eredeti pénzt (kap)! A legtöbb pénzt azok kapták (mindb vállalkozó), akiknél az osztások száma minimális, a legkevesebbet azok, akiknél az osztások száma maximális (maxdb vállalkozó).

```
Harmadol(n,kap,osztott,maxdb,mindb):
  kap(1):=0; osztott(1):=hamis
  Ciklus i=1-től n-ig
    Olvas(f,a,b,c)
    kap(b):=kap(a)+1; kap(c):=kap(a)+1; kap(a):=kap(a)+1
    osztott(a):=igaz; osztott(b):=hamis; osztott(c):=hamis
  Ciklus vége
  maxdb:=1
  Ciklus i=2-től 2*n+1-ig
    Ha kap(i)>kap(max) akkor max:=i
  Ciklus vége
  maxdb:=0
  Ciklus i=1-től 2*n+1-ig
    Ha kap(i)=kap(max) akkor maxdb:=maxdb+1
  Ciklus vége
  min:=1;
  Ciklus i=2-től 2*n+1-ig
    Ha kap(i)<kap(min) akkor min:=i
  Ciklus vége
  mindb:=0
  Ciklus i=1-től 2*n+1-ig
    ha kap(i)=kap(min) akkor mindb:=mindb+1
  Ciklus vége
Eljárás vége.
```

4. feladat: Konténer rendezés (15 pont)

Négyféle konténerünk lehet. Emiatt először le kell számlálnunk, hogy melyikből hány darab van!

Ezután felépítünk egy gráfot, amely megadja, hogy mely konténert mely másik helyére kell tenni. Hely(i) jelentse azt, hogy Hely(i-1)+1-től Hely(i)-ig kellene lenni az i-konténereknek! Minden i,j-re ($1 \le i,j \le 4$) jelölje At(i,j) az i-edik helyen lévő j-vel címkézett konténerek számát, tehát a bemeneti sorozatban azokan a j számoknak a számát, amelyek indexe Hely(i-1)+1 és Hely(i) közé esnek.

Tekintsük azt a G gráfot, amelynek 4 pontja van, 1,2,3 és 4, és i-ből j-be At(i,j) számú irányított él vezet!

Nyilvánvaló, hogy minden pontra a befutó élek száma megegyezik az onnan induló élek számával. Tehát minden összefüggő komponensre megadható olyan kör, amely minden élet (a multiplicitásának megfelelő számszor) tartalmaz (azaz Euler kör). Ha a körben lévő élek száma K, akkor K+1 lépésben helyükre rakhatók az elemek. Először egy tetszőleges elemet, ami nincs a helyén, legyen ez i, kivisszük a sor végén lévő szabad helyre, majd az így üresen maradt helyre a j elemet visszük, ha a körben $j \rightarrow i$ él van, és így tovább, amíg végigérünk a körön. Ekkor az üresen maradt helyre kell rakni az elsőként a szabad helyre kivitt elemet.

Tehát a szükséges lépések száma azon elemek száma, amelyek nincsenek a helyükön, plusz a G gráf komponenseinek száma. Belátható, hogy a G gráf egy vagy két komponensből állhat. Mivel csak a gráf komponenseinek számát kell meghatároznunk, ezért azt az irányított gráfot állítjuk elő, ahol $i \rightarrow j$ akkor és csak akkor él, ha $At(i,j)>0$.

Számítsuk ki minden pontnak a kifokát! Ha G két komponensből áll, akkor minden pont kifoka 1. Ha minden pont kifoka 1, akkor még állhat egy komponensből, nevezetesen, ha egyetlen kör. Ezt úgy tudjuk eldönteni, hogy ha $1 \rightarrow i$ él, akkor $i \rightarrow 1$ él-e?

```
Konténer(N,K,Mego):
  Hany():=0
  Ciklus i=1-től N-ig
    Hany(K(i)):=Hany(K(i))+1
  Ciklus vége
  Hely(0):=0
  Ciklus i=1-től 4-ig
    Hely(i):=Hely(i-1)+Hany(i)
  Ciklus vége
  G(,):=hamis; Mego:=0;
  Ciklus i=1-től N-ig
    Ciklus j=1-től 4-ig
      Ha K(i)≠j és Hely(j-1)+1≤i és i≤Hely(j)akkor
         Mego:=Mego+1; G(j,K(i)):=igaz
    Ciklus vége
  Ciklus vége
  Fok():=0
  Ciklus i=1-től 4-ig
    Ciklus j=1-től 4-ig
      Ha G(i,j) akkor Fok(i):=Fok(i)+1
    Ciklus vége
  Ciklus vége
  Ha Fok(1)=1 és Fok(2)=1 és Fok(3)=1 és Fok(4)=1 akkor
     j:=2
     Ciklus amíg nem G(1,j)
       j:=j+1
     Ciklus vége
     Ha G(j,1) akkor Plusz:=2 különben Plusz:=1
  különben Plusz:=1
  Ha Mego≠0 akkor Mego:=Mego+Plusz
Eljárás vége
```

5. feladat: Verem (15 pont)

Tároljuk, hogy melyik a kimeneten szereplő utolsó szám (`Sveg`)! Az i-edik szám a rendezett sorozatba kerülhet, ha ennél éppen eggyel nagyobb. Ha a verem tetején van az utolsónál eggyel nagyobb, akkor az kerülhet a rendezett sorozatba, különben pedig az `i`-edik szám mehet a verembe. Ha elfogytak a számok, akkor a veremből kiszedhetjük azokat, amelyek a sorozat végére illenek.

```
Verem(A,N,Sveg):
  Sveg:=0; i:=1; VeremInicializálás; Verembe(0)
  Ciklus amíg i≤N
    Ha A(i)=SVeg+1 akkor SVeg:=Sveg+1; i:=i+1
    különben ha Teteje=SVeg+1 akkor SVeg:=Sveg+1; Veremből
    különben Verembe(A(i)); i:=i+1
  Ciklus vége
  Ciklus amíg Teteje≠0 és Teteje=SVeg+1
    SVeg:=Sveg+1; Veremből
  Ciklus vége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Magánhangzók távolsága (10 pont)

Vegyünk fel egy tömböt a két- és háromjegyű mássalhangzók számára!

```
h: tömb(1..9,szöveg)
    =('cs','dz','gy','ly','ny','sz','ty','zs','dzs');
```

Haladjunk végig az s szöveg hangjain, ha egymás után nem két azonos mássalhangzót találunk, akkor számoljunk (db)! Ugyancsak számolni kell, ha többjegyű mássalhangzó után vagyunk (pl. sz után z következik).

Ha nem az első magánhangzóhoz érünk, akkor a számláló értékét ki kell írni, majd lenullázhatjuk.

```
Magánhangzók(s):
  db:=-maxint; i:=1; p:=igaz
  Ciklus amíg i≤hossz(s)-2
    Ha Magánhangzó(s(i)) akkor
       Ha db≥0 akkor Ír(g,db,' ')
       db:=0; i:=i+1
    különben
       Ha p vagy s(i)≠s(i-1) akkor db:=db+1
       i:=i+hossza(i)
  Ciklus vége
Eljárás vége.
```

Egy betű akkor magánhangzó, ha szerepel a magánhangzók halmazában.

```
Magánhangzó(c):
  Magánhangzó:=c∈{'a','á','e','é','i','í',
                  'o','ó','ö','ő','u','ú','ü','ű'}
Függvény vége.
```

Az i-edik karakteren kezdődő hang hosszát a h tömb alapján számoljuk.

```
hossza(i,p):
  p:=igaz
  Ha s(i)+s(i+1)+s(i+2)=h(9) akkor hossza:=3
  különben k:=1
           Ciklus amíg k≤8 és s(i)+s(i+1)≠h(k)
             k:=k+1
           Ciklus vége
           Ha k≤8 akkor hossza:=2
               különben hossza:=1; p:=hamis
Függvény vége.
```

2. feladat: Megrendelés (12 pont)

Először rendezzük a megrendeléseket a felső határuk szerint növekvő sorrendbe, az egyforma felső határúakat ezen belül az alsó határ szerint növekvő sorrendbe! Mivel a megrendelések sorszámára szükségünk lesz, ezért csak egy mutatótömböt (S) használunk a sorrend tárolására. A megoldás: az i-edik széket a Mego(i)-edik megrendelő kapja, ha ez nem 0. Ezt úgy számítjuk ki, hogy a rendezett megrendelés sorozaton haladunk végig, s minden megrendelést a lehető legkisebb hellyel próbálunk kielégíteni. Belátható, hogy ez a mohó megoldás helyes, ha ugyanis későbbi helyet adnánk, akkor ez esetleg megakadályozhatná egy későbbi megrendelés befejezését.

```
Megrendelés(N,A,B,Hány,Mego):
  S:=(1,2,...,N); Mego:=(0,...0)
  Rendez; Hány:=0
  Ciklus i=1-től N-ig
    ii:=S(i); x:=A(ii)
    Ciklus amíg x≤B(ii) és Mego(x)>0
      x:=x+1
    Ciklus vége
    Ha Mego(x)=0 akkor Mego(x):=ii; Hány:=Hány+1
  Ciklus vége
Eljárás vége.
```

3. feladat: Lámpák (16 pont)

Az első részfeladat megoldása nagyon egyszerű. Jelöljük `tér(i,j)`-vel azt, hogy az `(i,j)` mező világos-e! Kezdetben minden eleme legyen hamis, majd a lámpák beolvasásakor az adott négyzet alakú részeket igazra állítjuk. A végén a hamisan maradt mezők száma (`db`) lesz az első részfeladat megoldása.

```
Sötétek(k,tér,n,m,db):
  Ciklus ii=1-től k-ig
    Olvas(f,x,y)
    Ciklus i=x-l div 2-től x+l div 2-ig
      Ha i≥1 és i≤n akkor
        Ciklus j=y-l div 2-től y+l div 2-ig
          Ha j≥1 és j≤m akkor tér(i,j):=igaz
        Ciklus vége
    Ciklus vége
  Ciklus vége
  db:=0
  Ciklus i=1-től n-ig
    Ciklus j=1-től m-ig
      Ha nem tér(i,j) akkor db:=db+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A második részfeladat az (`1,1`) mezőről az (`n,m`) mezőre vezető leghosszabb út hosszát határozza meg, ahol az út hossza az általa érintett sötét mezők számát jelenti. Ezt egy szélességi gráfbejárással oldhatjuk meg, amelyben a szomszédos mezők közötti élek hossza akkor 1, ha az sötét mező, egyéb esetben pedig 0. A `táv` vektorban határozzuk meg a mezők (`1,1`) mezőtől vett távolságát, a megoldás a `táv(n,m)` értéke lesz. Kezdetben mindegyik eleme legyen -1 értékű!

```
Leghosszabb út(n,m,tér,táv):
  i:=1; j:=1
  Ha tér(i,j) akkor t:=0 különben t:=1
  PrsorInicializálás; Prsorba(i,j,t)
  Ciklus amíg táv(n,m)<0
    Prsorból(i,j,t)
    Ha i>1 és táv(i-1,j)<0 akkor
       Ha tér(i-1,j) akkor Prsorba(i-1,j,t)
                     különben Prsorba(i-1,j,t+1)
    Ha j>1 és táv(i,j-1)<0 akkor
       Ha tér(i,j-1) akkor Prsorba(i,j-1,t)
                     különben Prsorba(i,j-1,t+1)
    Ha i<n és táv(i+1,j)<0 akkor
       Ha tér(i+1,j) akkor Prsorba(i+1,j,t)
                     különben Prsorba(i+1,j,t+1)
    Ha j<m és táv(i,j+1)<0 akkor
       Ha tér(i,j+1) akkor Prsorba(i,j+1,t)
                     különben Prsorba(i,j+1,t+1)
  Ciklus vége
Eljárás vége.
```

4. feladat: Képkódolás (18 pont)

Első lépésként megnézzük, hogy az (`1,1`) pozíción kezdődő `n*n`-es kép egyszínű-e. Ha igen, akkor a kódját elő tudjuk állítani. Ha nem, akkor a képet felbontjuk 4 egyforma méretű részre, mindegyiknek előállítjuk a kódja következő karakterét, majd az adott résszel újra kezdjük a fenti eljárást.

Az előállított kódok száma (m) legyen globális változó!

```
Kódol(ks,ko,kn,kód,kép):
  Ha Egyszínű(ks,ko,kn) akkor kód(m):=kód(m)+'0'+kép(ks,ko)
  különben kn:=kn div 2; kkód:=kód(m)
           kód(m):=kkód+'1'; Kódol(m,ks,ko,kn)
           m:=m+1; kód(m):=kkód+'2'; Kódol(m,ks,ko+kn,kn)
           m:=m+1; kód(m):=kkód+'3'; Kódol(m,ks+kn,ko,kn)
           m:=m+1; kód(m):=kkód+'4'; Kódol(m,ks+kn,ko+kn,kn)
Eljárás vége.

Egyszínű(ks,ko,kn):
  i:=ks; j:=ko
  Ciklus amíg i<ks+kn és kép(i,j)=kép(ks,ko)
    Ha j<ko+kn-1 akkor j:=j+1 különben j:=ko; i:=i+1
  Ciklus vége
  Egyszínű:=(i=ks+kn)
Függvény vége.
```

5. feladat: Szavak (19 pont)

Nyilvánvaló, hogy ha van olyan betűhelyettesít, amely mellett az S1 és S2 szó képe megegyezik, akkor van olyan is, amikor minden betű helyére azozos betűből álló szót helyettesítünk. Legyen $a_1,\ldots a_k$ az S1-ben (ábácészerinti sorrendben) az egyes betűk előfordulási száma, $b_1,\ldots,b_k$ pedig az S2-beli előfordulási száma. Tehát az

$$a_1*x_1+\ldots+a_k*x_k = b_1*x_1+\ldots+b_k*x_k \quad (1)$$

egyenletet kell megoldani. Vagy másképpen írva

$$e1*x1+\ldots+ek*xk = 0 \quad (2)$$

ahol $e_i=b_i-a_i$H. Ha minden i-re $e_i\geq 0$ és legalább egy indexre $e_i>0$,vagy $e_i\leq 0$ és legalább egy indexre $e_i<0$, akkor nincs megpoldás, mert az ismeretlenek pozitív egész számok lehetnek, egyébként pedig van megoldás.

A megoldást az x_i ismeretlenek értékének lépésenként eggyel növelésével állítjuk elő. A kezdeti megoldáskezdemény a (bal, jobb) számpár, ahol bal az első egyenlet bal oldalának értéke, ha minden ismeretlen 1, jobb pedig az egyenlet jobb oldalának értéke. Az x_i ismeretlenek értékének eggyel növelésével a (bal+a_i, jobb+b_i) megoldáskezdeményt kapjuk. Világos, hogy csak a két oldal különbsége érdekes. Pontosabban, ha (bal1, jobb1) és (bal2, jobb2) két megoldáskezdemény és jobb1−bal1=jobb2−bal2, akkor csak azt a megoldáskezdeményt kell megtartani, amelyiknél az első komponens (bal1 vagy bal2) kisebb. Éppen ezért a megoldáskezdeményeket (bal, d, i) hármasként kezeljük, ami azt jelenti, hogy az egyenlet bal oldalának értéke bal, a jobb oldalé pedig bal+d, és egy másik megoldáskezdeményből úgy kaptuk, hogy a bal illetve jobb oldalhoz az a_i illetve a b_i együtthatót hozzáadtuk.

Megmutatjuk, hogy elég olyan megoldáskezdeményeket vizsgálni és tárolni, ahol a $|d|\leq L$, ahol L a maximuma az |L2−L1| és $|b_i-a_i|$ értékeknek (L1=hossz(S1), L2=hossz(S2)). Ennek belátásához a második formában írt egyenletet tekintsük. Ha van megoldás, akkor felírható egy

$$f_1+\ldots+f_u = 0 \quad (3)$$

összegzés, ahol minden $f_j\in\{e_1,\ldots,e_k\}$. Feltehetjük, hogy az első k tag éppen az $e_1,\ldots,e_k$együtthatók, mert minden ismeretlen csak pozitív egész lehet. Tehát ezek összege éppen L2−L1, amit jelöljünk d_0-lal.

Vegyük a $d_i=d_{i-1}+f_{k+i}$ sorozatot, $i=1,\ldots,u-k$! Azt állítjuk, hogy van olyan átrendezése a (3) egyenletben a tagoknak, hogy minden i-re $|d_i|\leq L$. Ez abból következik, hogy ha $d_{i-1}>0$, akkor biztosan tudunk f_{k+i} és f_u között olyan f_j–t választani, hogy $f_j<0$, úgyanis ha nem lenne ilyen,

akkor az összeg nem lehetne 0. Tehát ezt az f_j–t cseréljük fel f_{k+i}–vel. Hasonlóan járhatunk el, ha $f_{k+i}<0$.

Az aktív megoldáskezdeményeket tároljuk egy S sorban, amibe kezdetben az (L1, L2-L1)-at teszszük. Mivel egy d-re (jobb-bal különbségre) csak egy megoldáskezdeményt tárolunk, ezért a megoldáskezdemények azonosíthatók a d különbséggel. Az Ebal(d) az (1) egyenlet bal oldalának értékét tárolja, az X(d) pedig azt a karaktert, amelynek hozzáadásával kaptuk Ebal(d)-t. Ha adott d-re nincs még megoldáskezdemény, akkor X(d)='#'.

Az algoritmus hasonló a szélességi bejáráshoz. Amíg az S sor nem üres, kivesszük az első megoldáskezdeményt (d-t), majd minden karakterre megnézzük, hogy ha hozzáadjuk, akkor milyen megoldáskezdeményt kapunk. Ha az új megoldáskezdemény d-értéke |ujd|>L, akkor eldobjuk. Egyébként, ha még nem volt ilyen (azaz X(ujd)='#'), vagy volt, de az új bal oldal értéke kisebb, akkor betesszük a sorba az új megoldáskezdeményt. Egy megoldást az X vektorból tudunk előállítani visszafejtéssel.

```
Előkészítés(S1,S2,A,B,L):
  Ciklus c='A'-tól 'Z'-ig
    A(c):=0; B(c)2:=0
  Ciklus vége
  L1:=hossz(S1); L2:=hossz(S2);
  Ciklus i=1-től L1-ig
    inc(A((S1(i))
  Ciklus vége
  Ciklus i=1-től L2-ig
    inc(A(S2(i))
  Ciklus vége
  Poz:=0; Neg:=0; L:=0;
  Ciklus c='A'-tól 'Z'-ig
    if |A(c)-B(c)|>L akkor L:= |A(c)-B(c)|
    Ha A(c)-B(c)>0 akkor inc(Poz);
    Ha A(c)-B(c)<0 akkor inc(Neg);
  Ciklus vége
  Ha L1>L akkor L:=L1;
  Ha Poz>0 és Neg=0 vagy Poz=0 és Neg>0
     akkor X(0)='#'; Kiír(A,B,X) {nincs megoldás}
  különben Megold(A,B,X)
Eljárás vége.
```

```
Megold(A,B,X):
  X(L2-L1):='*'; d:=L2-L1; Ebal(d):=L1
  eleje:=1; vege:=1; {sor inicializálás}
  SorBa(d);
  Ciklus amíg NemUres
    d:=SorBol;
    Ciklus c='A'-tól'Z'-ig
      Ha (A(c)>0)or (B(c)>0) akkor
        bal:=Ebal(d)+A(c); ujd:=d+B(c)-A(c);
        Ha |ujd|<=L és X(ujd)='#' vagy bal<Ebal(ujd)
            akkor X(ujd):=c; Ebal(ujd):=bal; SorBa(ujd)
      Ciklus vége;
  Ciklus vége
  Kiír(A,B,X)
Eljárás vége.
```

```
Kiír(A,B,X):
  Ha X(0)='#' akkor Ír(Ki,0) {nincs megoldás}
  különben
    d:=0; M:=L1;
    Ciklus c='A'-tól 'Z'-ig
      Ha A(c)>0 vagy B(c)>0 akkor H(c):=1
                            különben H(c):=0
    Ciklus vége
    Ciklus amíg X(d)<>'*' {megoldás visszafejtés}
      c:=X(d); H(c):=H(c)+1; M:=M+A(c); d:=d-(B(c)-A(c))
    Ciklus vége
    Ki(KiF,M)
    Ciklus c='A'-től 'Z'-ig
      Ha H(c)>0
        Ki(Ki,c,' ')
        Ciklus i=1-től H(c)-ig
          Ír(Ki,'A')
        Ciklus vége
        Újsor(Ki)
      Ciklus vége
    Ciklus vége
Eljárás vége
```

2003. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Naptár (24 pont)

Tároljuk az egyes hónapok napszámát a `h` vektorban, amiben a februárt szökőévben át kell írni 29-re!

```
h: tömb(1..12,egész)=(31,28,31,30,31,30,31,31,30,31,30,31)
```

Az 1582. október 4. előtti dátumokat nem kell megváltoztatni, a későbbiekhez viszont meg kell növelni. Egyrészt minden 10 nappal eltolódik, másrészt arra is figyelni kell, hogy a Gergely naptárban azonban a 100-zal osztható évek közül csak a 400-zal is oszthatók szökőévek.

```
Naptár(év,hó,nap,gév,ghó,gnap):
  Ha év<1582 vagy év=1582 és (hó<10 vagy hó=10 és nap≤4)
     akkor gév:=év; ghó:=hó; gnap:=nap
  különben kül:=10+(év-1600) div 100-(év-1600) div 400
            Ha hó≤2 és (év mod 100)=0 (év mod 400)>0
               akkor kül:=kül-1
            gnap:=nap+kül; ghó:=hó; gév:=év
            Ha (év mod 400)=0 vagy
               (év mod 100)>0 és (év mod 4)=0
               akkor ha hó=2 akkor h(hó):=29
            Ha gnap>h(hó)
               akkor gnap:=gnap-h(hó); ghó:=ghó+1
                     Ha ghó>12 akkor gév:=gév+1; ghó:=1
  Elágazások vége
Eljárás vége.
```

2. feladat: Verseny (25 pont)

Jelölje `v(i)=igaz`, ha az i-edik versenyző célba ért! Kezdetben a `v` vektor minden elem legyen hamis, amit célba érésnél állítunk igazra! A célba ért versenyzők futási idejét a beérkezési időből és a rajtszámból számolhatjuk ki.

Ezután rendezni kell futási idő szerint, majd jöhet a kiírás, de figyelni kell a holtversenyre is!

```
Verseny(n,m):
  v(i):=(hamis, ..., hamis)
  Ciklus i=1-től m-ig
    Be: t(i).sor; v(t(i).sor):=igaz
    Be: j; t(i).idő:=j-t(i).sor
  Ciklus vége
  Ciklus i=1-től m-1-ig
    min:=i
    Ciklus j=i+1-től m-ig
      Ha t(j).idő<t(min).idő akkor min:=j
    Ciklus vége
    s:=t(i); t(i):=t(min); t(min):=s
  Ciklus vége
  j:=t(1).idő-1
  Ciklus i=1-től m-ig
    Ha j<t(i).idő akkor Ki: i,'. helyezett:'
    Ki: t(i).sor; j:=t(i).idő
  Ciklus vége
  Ha m<n akkor Ki: 'Nem érkezett célba:'
               Ciklus i=1-től n-ig
                 Ha nem v(i) akkor Ki: i
               Ciklus vége
Eljárás vége.
```

3. feladat: Kártya (26 pont)

Tároljuk a francia kártya lehetséges színeit és értékeit egy-egy tömbben!

```
szín: tömb(1..4,szöveg)=('treff','pikk','kör','káró')
érték: tömb(1..13,szöveg)=('2','3','4','5','6','7','8','9',
                           '10','bubi','dáma','király','ász')
```

Legyen a `t(i,j)`=igaz, ha az i-edik szín j-edik lapja a játékosnál van! Ezt beolvasásnál a fenti két tömbben kereséssel kitölthetjük. Ezután megnézzük minden színre, hogy van-e a játékosnál három egymás utáni kártya, majd pedig, hogy ugyanaz az értékű kártya van-e nála három színből.

```
Kártya(t):
  Ciklus i=1-től 4-ig
    Ciklus j=1-től 11-ig
      Ha t(i,j) és t(i,j+1) és t(i,j+2) akkor
        Ki: szín(i),érték(j),érték(j+1),érték(j+2)
    Ciklus vége
  Ciklus vége
  Ciklus j=1-től 13-ig
    Ha t(1,j) és t(2,j) és t(3,j) akkor
      Ki: szín(1),érték(j),szín(2),érték(j),szín(3),érték(j)
    Ha t(1,j) és t(2,j) és t(4,j) akkor
      Ki: szín(1),érték(j),szín(2),érték(j),szín(4),érték(j)
    Ha t(1,j) és t(3,j) és t(4,j) akkor
      Ki: szín(1),érték(j),szín(3),érték(j),szín(4),érték(j)
    Ha t(2,j) és t(3,j) és t(4,j) akkor
      Ki: szín(2),érték(j),szín(3),érték(j),szín(4),érték(j)
  Ciklus vége
Eljárás vége
```

Kilencedik-tizedik osztályosok

1. feladat: Barátok (15 pont)

Vegyük kezdetben mindegyik tanulót külön halmazba! Ha két tanuló barátja egymásnak, akkor az őket tartalmazó halmazokat össze lehet vonni. Az egyes halmazokat a legkisebb sorszámú tagjukkal reprezentálhatjuk a b vektorban.

```
Barátok(n,m,db,b):
  db:=N
  Ciklus i=1-től n-ig
    b(i):=i
  Ciklus vége
  Ciklus i=1-től m-ig
    Olvas(BeF,j,k)
    Ha b(j)≠b(k) akkor
      Ha b(j)>b(k) akkor Csere(b(j),b(k))
      bx:=b(k); db:=db-1
      Ciklus k=1-től n-ig
        Ha bx=b(k) akkor b(k):=b(j)
      Ciklus vége
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

2. feladat: Szállítás (15 pont)

Előkészítésként a telephelyek gráfjának minden pontjára határozzuk meg, hogy hány be- (be), illetve kimenő (ki) éle van!

```
Előkészítés(n,m,be,ki,s):
  be:=0; ki:=0
  Ciklus i=1-től m-ig
    Olvas(BeF,j,k); ki(j):=ki(j)+1; be(k):=be(k)+1
    s(i,1):=j; s(i,2):=k
  Ciklus vége
Eljárás vége.
```

A csak termeléssel foglalkozó telephelyek azok, amelyeknek nincs bemenő éle.

```
SzállítA(n,be):
  db:=0
  Ciklus i=1-től n-ig
    Ha be(i)=0 akkor db:=db+1
  Ciklus vége
  Ír(KiF,db)
  Ciklus i=1-től n-ig
    Ha be(i)=0 akkor Ír(KiF,' ',i)
  Ciklus vége
Eljárás vége.
```

A csak árusítással foglalkozó telephelyek azok, amelyeknek nincs kimenő éle.

```
SzállítB(n,be):
  db:=0
  Ciklus i=1-től n-ig
    Ha ki(i)=0 akkor db:=db+1
  Ciklus vége
  Ír(KiF,db)
  Ciklus i=1-től n-ig
    Ha ki(i)=0 akkor Ír(KiF,' ',i)
  Ciklus vége
Eljárás vége.
```

Azok az árusító telephelyek, ahova csak termelő telephelyről küldenek árut olyanok, hogy a bemenő éleik végpontjaiba nem vezet él.

```
SzállítC(n,n,be,ki,s):
  db:=0
  Ciklus i=1-től n-ig
    Ha ki(i)=0 és Jó(i,m,be,s) akkor db:=db+1
  Ciklus vége
  Ír(KiF,db)
  Ciklus i=1-től n-ig
    Ha ki(i)=0 és Jó(i,m,be,s) akkor Ír(KiF,' ',i)
  Ciklus vége
Eljárás vége.

Függvény Jó(i,m,be,s):
  j:=1
  Ciklus amíg j≤m és nem (s(j,2)=i és be(s(j,1))>0)
    j:=j+1
  Ciklus vége
  Jó:=(j>m)
Függvény vége.
```

A raktározó telephelyek, akiknek nincs kapcsolatuk közvetlenül sem termelővel, sem árusítóval olyanok, hogy van legalább egy be- és kimenő élük is, továbbá egyik bemenő élük sem jön termelő csomópontból, és egyik kimenő élük sem megy árusító csomópontba.

```
SzállítD(n,m,be,ki,s):
  db:=0
  Ciklus i=1-től n-ig
    Ha be(i)≥1 és ki(i)≥1 és Jó(i,m,be,ki,s) akkor db:=db+1
  Ciklus vége
  Ír(KiF,db)
  Ciklus i=1-től n-ig
    Ha be(i)≥1 és ki(i)≥1 és Jó(i,m,be,ki,s) akkor Ír(KiF,i)
  Ciklus vége
Eljárás vége.

Függvény Jó(i,m,be,ki,s):
  j:=1
  Ciklus amíg j≤m és nem (s(j,2)=i és be(s(j,1))=0 vagy
                          s(j,1)=i és ki(s(j,2))=0)
    j:=j+1
  Ciklus vége
  Jó:=(j>m)
Függvény vége.
```

Sem összegyűjtő, sem szétosztó funkciója nincs azoknak a raktározó telephelyeknek, amelyeknek pontosan egy be- és egy kimenő éle van.

```
SzállítE(n,be,ki):
  db:=0
  Ciklus i=1-től n-ig
    Ha be(i)=1 és ki(i)=1 akkor db:=db+1
  Ciklus vége
  Ír(KiF,db)
  Ciklus i=1-től n-ig
    Ha be(i)=1 és ki(i)=1 akkor Ír(KiF,' ',i)
  Ciklus vége
Eljárás vége.
```

3. feladat: Titkos társaság (15 pont)

Az adatok beolvasásánál minden tagról tároljuk, hogy ki van felette (`felette`) a társaság hierarchiájában.

Az A részfeladat szerint egy ember akkor küldhet levelet egy másiknak, ha a hierarchiában felette van.

```
Küldhet(a,b):
  Ciklus amíg felette(b)≠0 és b≠a
    b:=felette(b)
  Ciklus vége
  Küldhet:=(b=a)
Függvény vége.
```

A B részfeladat szerint a két ember távolsága (`tav`) a lépések száma, ha egyik ember közvetve felettese a másiknak, különben pedig a legközelebbi közös felettesüktől vett távolságok összege.

```
Távolság(a,b):
  c:=a
  Ciklus amíg felette(a)≠0 és a≠b
      tav(felette(a)):=tav(a)+1; a:=felette(a)
  Ciklus vége
  Ha a=b akkor Távolság:=tav(b)
      különben a:=c
               Ciklus amíg a≠b és tav(felette(b))=0
                 tav(felette(b)):=tav(b)+1; b:=felette(b)
               Ciklus vége
               Ha a=b akkor Távolság:=tav(a)
               különben Távolság:=tav(felette(b))+tav(b)+1
Függvény vége.
```

Egy ember összes beosztottja a közvetlen beosztottjai, valamint azok összes beosztottja.

```
Beosztottak(a):
  s:=0
  Ciklus i=1-től n-ig
    Ha felette(i)=a akkor s:=s+1+beosztottak(i)
  Ciklus vége
  Beosztottak:=s
Függvény vége.
```

4. feladat: Rendőrök (15 pont)

Minden i-re ($1 \le i \le N$) és minden j-re ($0 \le j \le N$) tekintsük azt a részproblémát, hogy minimálisan mekkora költséggel lehet elérni, hogy az első i város mindegyikében legyen legalább egy rendőr, és az i-

edik városban még legyen legalább plusz j rendőr, de csak az első i városban lévő rendőröket mozgathatjuk!

Jelölje M(i,j) ezt az értéket, ami legyen végtelen, ha j-re ez nem teljesíthető, mert nincs annyi rendőr az első i városban! Továbbá, tekintsük azt a részproblémát, hogy minimálisan mekkora költséggel érhető el az, hogy az első i város mindegyikében legyen legalább egy rendőr, ha j ($1 \le j \le N$) rendőrt kölcsönvennénk az i+1-edik városból! Jelölje ezt az értéket M(i,-j), ami ismét végtelen, ha nem lehet ilyen megoldás! Nyilvánvaló, hogy a feladat megoldása M(N,0). M(i,j)-t definiáljuk i=0-ra is, $M(0,0)=0$ és $M(0,j)=\infty$ egyébként. M(i,j) kiszámítására rekurzív összefüggés adható, ha i>0, akkor M(i,j) kiszámítható M(i-1,k), k=-N,…,N értékeivel.

Mivel csak az optimális megoldás értékét kell kiszámítani, alkalmas kiszámítási sorrendet alkalmazva, egydimenziós tömb is elég.

```
Rendőrök(N,R):
  Hany(0):=0
  Ciklus i=1-től N-ig
    Hany(i):=Hany(i-1)+R(i); M(-i):=Inf; M(i):=Inf
  Ciklus vége
  M(0):=0
  Ciklus i=1-től N-ig
    k:=R(i); Plusz:=Hany(i)-i
    Ha Plusz>N akkor Plusz:=N
    Ha Plusz<0 akkor Plusz:=0
    Hiany:=Hany(N)-Hany(i-1)-(N-i+1)
    Ha Hiany>N akkor Hiany:=N
    Ha Hiany<0 akkor Hiany:=0
    Ha k=0 akkor Ciklus j=-N-től N-1-ig
                   M(j):=M(j+1)+|j+1|
                 Ciklus vége
    különben ha k=1 akkor Ciklus j=-N-től N-ig
                            M(j):=M(j)+|j|
                          Ciklus vége
    különben {k>1} Ciklus j=k-tól Plusz-ig
                     M(j):=M(j-k+1)+j-k+1
                   Ciklus vége
    M(k-1):=M(0)
    Ciklus j=k-2-től 0-ig -1-esével
      Ha M(j+1)< M(j+1-k)+k-j-1 akkor M(j):=M(j+1)
                                különben M(j):=M(j+1-k)+k-j-1
    Ciklus vége
    Ciklus j=Plusz+1-től N-ig
      M(j):=Inf
    Ciklus vége
    Ciklus j=1-től Hiany-ig
      M(-j):=M(-j-(k-1))+k-1
    Ciklus vége
    Ciklus j=Hiany+1-től N-ig
      M(-j):=Inf
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

5. feladat: Futó (15 pont)

Ha ismerjük a futók távolságát a céltól (`t`), valamint az 1 másodperc alatt megtett útjukat (`a`), akkor kiszámolhatjuk, hogy menyi idő alatt érnek célba (`idő(i):=t(i)/ a(i)`).

Mivel az adatokat távolság szerint csökkenő sorrendben kapjuk, ezért csak azt kell megnézni, hogy egy kisebb sorszámú futó (távolabbról indult) lehagy-e egy nagyobb sorszámút (közelebbről indultat). Ha igen, akkor ki kell számolni a lehagyás idejét, majd be kell szúrni az ezekből készült rendezett sorozatba!

```
Futó(n,idő,t,a,db,x):
  db:=0
  Ciklus i=1-től n-1-ig
    Ciklus j=i+1-től n-ig     {i megelőzi-e j-t?}
      Ha idő(i)<idő(j)
         akkor k:=((t(i)-t(j)) div (a(i)-a(j)))+1
                  {ha k ∉ a kigyűjtötteknek, akkor beszúrjuk}
               e:=1; u:=db; koz:=(e+u) div 2
               Ciklus amíg e≤u és x(koz)≠k
                 Ha x(koz)<k akkor e:=koz+1
                           különben u:=koz-1
                 koz:=(e+u) div 2
               Ciklus vége
               Ha e>u
                  akkor db:=db+1
                        Ciklus u=db-től e+1-ig -1-esével
                          x(u):=x(u-1)
                        Ciklus vége
                        x(e):=k
      Elágazás vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Régió (15 pont)

Vegyük kezdetben mindegyik települést külön halmazba, azaz régióba! Ha két település távolsága kisebb T-nél, akkor az őket tartalmazó halmazokat össze lehet vonni. Az egyes halmazokat egyik tagjukkal reprezentálhatjuk a b vektorban.

```
Régió(n,m,T,h,db,b):
  db:=N
  Ciklus i=1-től n-ig
    b(i):=i
  Ciklus vége
  Ciklus i=1-től m-ig
    Olvas(BeF,j,k)
    Ha b(j)≠b(k) akkor
      Ha |h(j,1)-h(k,1)|+|h(j,2)-h(k,2)|≤T
         akkor bx:=b(k); db:=db-1
               Ciklus k=1-től n-ig
                 Ha bx=b(k) akkor b(k):=b(j)
               Ciklus vége
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

2. feladat: Repülőút (16 pont)

Előkészítésként minden városra számoljuk ki a beérkező járatok számát (befok), valamint a kimenő járatok számát (kifok)! A városokon a sorszámok szerinti sorrendben haladunk. Nyilvántartjuk az minden pillanatban a kilépő és a belépő élek számának különbségét.

Ha az aktuális pontnál levonjuk ebből a számból az ő belépő élei számát, akkor pontosan akkor kapunk 0-át, ha a város kikerülhetetlen (azaz nincs olyan él, ami nála kisebb sorszámú pontból nála nagyobb sorszámú pontba vezetne). Ha egy ilyen pontból egyetlen kimenő él megy tovább, akkor az az él (járat) kikerülhetetlen.

```
Repülőút(n,befok,kifok,db1,x,db2,y):
  db:=0; db1:=0; db2:=0
  Ciklus i=1-től n-1-ig
    db:=db-befok(i)
    Ha db=0 akkor db1:=db1+1; x(db1):=i
                  Ha kifok(i)=1 akkor db2:=db2+1; y(db2):=i
    db:=db+kifok(i)
  Ciklus vége
Eljárás vége.
```

3. feladat: Titkos társaság (15 pont)

A társaság egy bináris fával írható le. Legyen `binfa(i,1)` az i-edik ember egyik beosztottja sorszáma, `binfa(i,2)` pedig a másik beosztottjáé! 0-val jelöljük, ha valamelyik nincs. `Binfa(i,3)` pedig az i-edik ember főnökének a sorszáma legyen!

Az A részfeladat szerint egy ember összes beosztottja a közvetlen beosztottjai, valamint azok összes beosztottja.

```
Beosztottak(a):
  s:=0
  Ha binfa(a,1)>0 akkor s:=s+1+beosztottak(binfa(a,1))
  Ha binfa(a,2)>0 akkor s:=s+1+beosztottak(binfa(a,2))
  Beosztottak:=s
Függvény vége.
```

A B részfeladat szerint a két ember távolsága (`tav`) a lépések száma, ha egyik ember közvetve felettese a másiknak, különben pedig a legközelebbi közös felettesüktől vett távolságok összege.

```
Távolság(a,b):
  c:=a
  Ciklus amíg binfa(a,3)≠0 és a≠b
    tav(binfa(a,3)):=tav(a)+1; a:=binfa(a,3)
  Ciklus vége
  Ha a=b akkor Távolság:=tav(b)
      különben a:=c
               Ciklus amíg a≠b és tav(binfa(b,3))=0
                 tav(binfa(b,3)):=tav(b)+1; b:=binfa(b,3)
               Ciklus vége
               Ha a=b akkor Távolság:=tav(a)
               különben Távolság:=tav(binfa(b,3))+tav(b)+1
  Elágazás vége
Eljárás vége.
```

A C részfeladat megoldásához először számítsuk ki minden pontra az abból a pontból induló fa magasságát! Ez rekurzívan egy menetben kiszámolható.

Ha egy embernek nincs első beosztottja, akkor a belőle induló fa 0 magasságú. Ha egy beosztottja van, akkor a magassága a beosztottjából induló fa magasságánál eggyel nagyobb. Ha két beosztottja van, akkor a magassága a belőlük induló magasabb fa magasságánál eggyel nagyobb.

```
Magasságok(i):
  Ha binfa(i,1)=0 akkor mag(i):=0
  különben ha binfa(i,2)=0
             akkor Magasságok(binfa(i,1));
                   mag(i):=mag(binfa(i,1))+1
  különben Magasságok(binfa(i,1)); Magasságok(binfa(i,2))
           mag(i):=max(mag(binfa(i,1)),mag(binfa(i,2)))+1
Eljárás vége.
```

Ezután a C részfeladat megoldása szintén rekurzívan számolható. Az i-edik emberrel kezdődő fa leghosszabb útja 0 hosszúságú, ha nincs egyetlen beosztottja sem. Ha egyetlen beosztottja van, akkor a leghosszabb út vagy a belőle kiinduló részfa magassága, vagy pedig a beosztottjával kezdődő részfa leghosszabb útja. Ha két beosztottja van, akkor a leghosszabb út vagy a két részfa magasságának összege plusz 2, vagy valamelyik részfa leghosszabb útja.

```
Maxtáv(i):
  Ha binfa(i,1)=0 akkor Maxtáv:=0
  különben ha binfa(i,2)=0
             akkor Maxtáv:=max(mag(i),Maxtáv(binfa(i,1)))
  különben Maxtáv:=max(Maxtáv(binfa(i,1)),
                       Maxtáv(binfa(i,2)),
                       mag(binfa(i,1))+mag(binfa(i,2))+2)
Függvény vége.
```

4. feladat: Rendőrök (14 pont)

Mivel legfeljebb annyi rendőr van, mint város, ezért az olyan elrendezés, amelyben a legtöbb városban van rendőr azt jelenti, hogy sehol sincs egynél több rendőr.

Tekintsünk egy optimális megoldást, amely $i \to j$ indexpárok formájában megadja, hogy honnan hova kell átmozgatni egyesével! Az átmozgatások nem keresztezhetik egymást, tehát nem lehet olyan $i_1 \to j_1$ és $i_2 \to j_2$ átmozgatás, hogy $i_1 < j_2 < j_1 < i_2$, mert ekkor $i_1 \to j_2$ és $i_2 \to j_1$ kevesebb lépést igényelne. Minden olyan $1 \le i \le j \le N$ indexpárra, amelyre teljesül, hogy `i`-től `j`-ig legfeljebb `j-i+1` rendőr van, határozzuk meg, hogy legkevesebb hány lépéssel lehet elérni, hogy sehol se legyen egynél több rendőr. Jelölje ezt az értéket `E(i,j)`!

```
Rendőrök(N,R):
  Hany(0):=0
  Ciklus i=1-től N-ig
    Hany(i):=R(i)+Hany(i-1)
    Ciklus j=i-től N-ig
      T(i,j):=-1
    Ciklus vége
  Ciklus vége
  Eloszt(1,N);
  Bejár(1,N) {egy optimális elrendezés kiíratása}
Eljárás vége.
```

Ha `i`-től `j`-ig pontosan `j-i+1` rendőr van, akkor egyszerű mohó módszerrel kiszámítható `E(i,j)` értéke, ezt végzi az Egyenget eljárás. Egyébként

$$E(i,j) = \min\left(E(i,k) + E(k+1,j)\right)$$

ahol $i \le k < j$ és `i`-től `k`-ig legfeljebb `k-i+1` rendőr van és `k+1`-től `j`-ig legfeljebb `j-k` rendőr van. Nyilvánvaló, hogy ha `i`-től `j`-ig nincs egy rendőr sem, akkor `E(i,j)=0`. A kiszámítást rekurzió-memórizálás módszerével végzi az Eloszt eljárás. A `Közép(i,j)` tömbelem tárolja azt a `k` értéket, amelyre a képletben a minimum adódik. Pontosabban, `Közép(i,j)=j`, ha `i`-től `j`-ig `j-i+1` rendőr van. Ezt felhasználva tudunk egy optimális elrendezést kiíratni, amit a Bejár rekurzív eljárás végez el.

```
Eloszt(i,j):
  Ha T(i,j)≥0 akkor Eloszt:=T(i,j)
  különben
    Ha Hany(j)-Hany(i-1)=0 akkor Minlep:=0; kKozep:=0
    különben ha Hany(j)-Hany(i-1)=j-i+1
                akkor Minlep:=Egyenget(i,j); Kkozep:=j
    különben
      Minlep:=N*N
      Ciklus k=i-től j-1-ig
        Ha Hany(k)-Hany(i-1)≤k-i+1 és Hany(j)-Hany(k)≤j-k
           akkor lep:=Eloszt(i,k)+Eloszt(k+1,j)
                 Ha lep<Minlep akkor Minlep:=lep; kkozep:=k
      Ciklus vége
    T(i,j):=Minlep; Kozep(i,j):=kkozep; Eloszt:=Minlep
  Elágazás vége
Eljárás vége.
```

Mivel az (i,j) szakaszon pontosan j-i+1 rendőr van, ezért minden helyre, ahol R(k)=0, egy rendőrt kell vinni olyan helyről, ahol egynél több rendőr van. Legyen üres a legkisebb olyan index (i≤üres≤j), hogy R(üres)=0, és több a legkisebb olyan index, (i≤több≤j), hogy R(több)>1. Ekkor van olyan optimális átrendezés, amelyben a több helyről mozgatunk át az üres helyre. Ezt ismételve amíg van üres hely, megkapjuk a megoldást.

```
Egyenget(i,j):
  lép:=0; üres:=i-1; több:=i-1; RR:=0
  Ciklus
    Ciklus
      üres:=üres+1
    amíg üres≤j és R(üres)≠0
    Ciklus vége
    Ha üres≤j akkor
       Ha RR≤1 akkor Ciklus
                       több:=több+1
                     amíg több≤j és R(több)≤1
                     Ciklus vége
                     Ha több≤j akkor RR:=R(több)
       Ha több≤j akkor lép:=lép+Abs(üres-több); RR:=RR-1
  amíg üres≤j és több≤j
  Ciklus vége
  Egyenget:=lép
Eljárás vége.
```

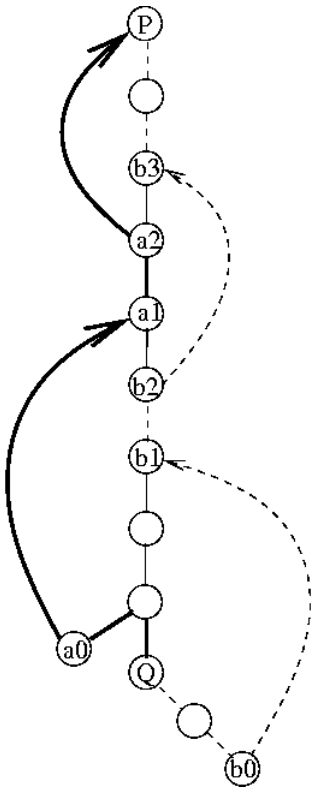
```
Bejár(i,j):
  Ha Közép(i,j)>0
     akkor Ha Közép(i,j)=j
              akkor Ciklus k=i-től j-ig
                      RR(k):=1
                    Ciklus vége
           különben Bejár(i,Közép(i,j))
                    Bejár(Közép(i,j)+1,j)
Eljárás vége.
```

5. feladat: Hálózat (15 pont)

Mivel bármely két csomópont között van két, közös pontot nem tartalmazó útvonal, ezért a gráfra teljesül, hogy bármely élét elhagyva a gráf összefüggő marad.

Vegyünk egy P-gyökerű mélységi feszítőfát! A Q pontnak biztosan van olyan b0 leszármazottja a feszítőfában, amelyből van visszaél Q valamely ősébe. Ez azért igaz, mert egyébként a Q apjából

q-ba vezető feszítőfa élet törölve a gráf nem lenne összefüggő. Vegyük a legmagasabbra, (azaz a legkisebb elérési idejű) visszamutató visszaél végpontját, az ábrán ez a b1 pont. A b1 pontnak van olyan a0 leszármazottja, amelyből van visszaél b1 valamely ősébe, legyen ez az a1 pont. Az a0 nem lehet a b1, mert akkor b1-nek négy szomszédja lenne, de a bemeneti feltétel szerint legfeljebb három lehet csak. Ha több ilyen van, akkor vegyük a legkisebb elérésű idejű pontba visszamutató élet. Az a0 pont biztosan nem leszármazottja Q-nak a feszítőfában b1 választása miatt.



Ezt az eljárást folytassuk addig, amíg P-be visszamutató élet nem kapunk. Ekkor az ábrán látható vastagabb vonallal jelzett P-a2-a1-a0-Q és a szaggatott vonallal jelzett P-b2-b2-b1-b0-Q két diszjunkt út lesz P és Q között.

Hogyan lehet meghatározni (hatékonyan) a kívánatos visszaéleket? Először is a mélységi bejárás során számítsuk ki minden u pontnak az elérési idejét, legyen ez E(u). Továbbá, legyen Ős(u) az a w pont, amelyhez van olyan v leszármazottja (vagy maga u) u-nak, hogy v→w visszaél és E(w) a legkisebb. Ez a rekurzív mélységi bejárással kiszámítható. Azonban tudnunk kell a feszítőfában az u-ból v-be vezető utat is. Ezért a Fel(u) értéke legyen a feszítőfában az u-ból v-be vezető út első pontja. Ezt is a mélységi bejárás során számítjuk. A MélyBejár által kiszámított Apa, Ős, és Fel függvényekkel az Utaz1 az egyik, az Utaz2 a másik utat határozza meg.

```
Hálózat(N,A,B,P1,P1n,P2,P2n):
  Apa:=0; Apa(A):=A; Os(A):=A; Idő:=0
  MélyBejár(A)  {a mélységi feszítőfa építése, Apa,Os,Fel}
  P1n:=1; P1(1):=B; P2n:=1; P2(1):=B
  Utaz1(B, P1, P1n)    {egy B→A út előállítása}
  Utaz2(B, P2, P2n)    {}
Eljárás vége.

MélyBejár(u):
  E(u):=Idő; Idő:=Idő+1; w:=u; bw:=u; j:=1; v:=G(u,j)
  Ciklus amíg v>0                {u→v élen vizsgáljuk}
    Ha Apa(v)=0 akkor            {u→v faél}
       Apa(v):=u; MélyBejár(v)
       Ha E(Os(v))<E(w) akkor w:=Os(v); bw:=v
    különben             {u→v visszaél}
       Ha v≠Apa(u) és E(v)<E(w) akkor w:=v; bw:=u
    j:=j+1
    Ha j≤3 akkor v:=G(u,j) különben v:=0
  Ciklus vége
  Os(u):=w
  Ha bw=u akkor Fel(u):=w különben Fel(u):=bw
Eljárás vége.
```

```
Utaz1(x,P,Pn):
  Os1:=Os(x); Os2:=Os(Os1)
  Ciklus amíg x≠A
    Ciklus
      x:=Fel(x); Pn:=Pn+1; P(Pn):=x
    amíg x≠Os1
    Ciklus vége
    Ciklus amíg x≠Os2 és Os(x)=Os2
      x:=Apa(x); Pn:=Pn+1; P(Pn):=x
    Ciklus vége
    Os1:=Os(x); Os2:=Os(Os1)
  Ciklus vége
Eljárás vége.
```

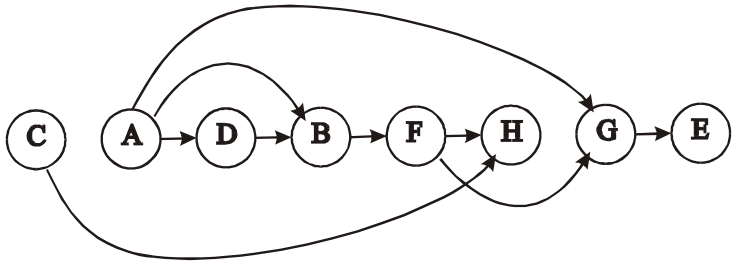
```
Utaz2(x,P,Pn):
  Os1:=Os(x); Os2:=Os(Os1)
  Ha Os1=Os2 akkor Ciklus amíg x≠Os2
                      x:=Apa(x); Pn:=Pn+1; P(Pn):=x
                    Ciklus vége
  különben Ciklus amíg Os(x)≠Os2
             x:=Apa(x); Pn:=Pn+1; P(Pn):=x
           Ciklus vége
  Ha x≠Os2 akkor Utaz1(x,P,Pn)
Eljárás vége.
```

2004. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendezés (30 pont)

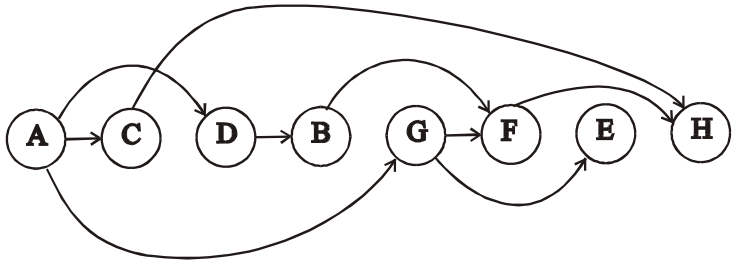
A. C,A,D,B,F,H,G,E 5 pont



(C) a (H) előtt bárhol lehet, (H) a (G,E)-hez képest tetszőleges helyen lehet, (H) és (G) sorrendje mindegy.

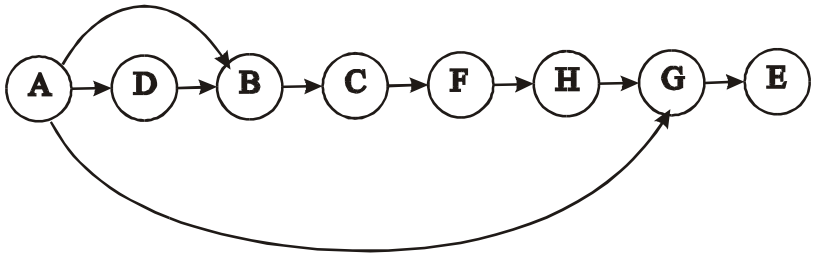
B. (A,B,F,G,A) körbeverés vagy (A,D,B,F,G,A) körbeverés 5 pont

C. A,C,D,B,G,F,E,H 5 pont



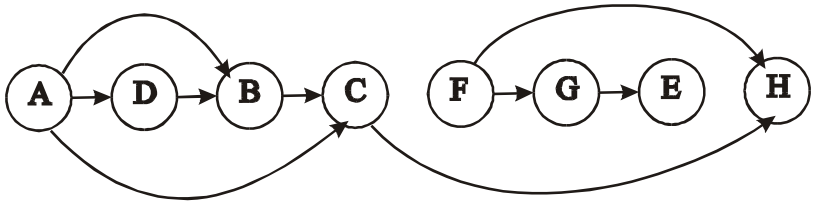
(D,G) és (B,G) sorrendje mindegy, (E) a (G) után bárhol lehet, (H) az (F) után bárhol lehet, (C) az (A) és a (H) között bárhol lehet.

D. csak az (A,D,B,C,F,H,G,E) sorrend jó 5 pont



E. (A,B,F,H,A) körbeverés. (A,D,B,F,H,A) is 5 pont

F. A,D,B,C,F,G,E,H 5 pont



(H) az (F) után bárhol lehet, (A,D,B,C) a (H) előtt bárhol lehet.

Hibás megoldásokra annyi pontot kell levonni, ahány sorrendi szabályt sért, de 0 pontnál kevesebb nem adható (azaz például az F részfeladatnál a D,A,B,C,F,G,E,H megoldásra 4 pont jár)!

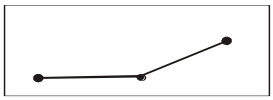
2. feladat: Gépi nyelv (21 pont)

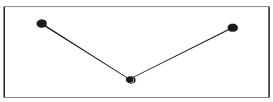
```
ADD A,A         ADD A,A         MOV A,B
ADD A,A         MOV A,B         ADD A,A
ADD A,A         ADD A,A         ADD A,B
ADD A,A         ADD A,A         ADD A,A
                ADD A,B         ADD A,A
                ADD A,B
```

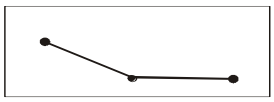
Programonként 6-7-8 pont.

Más helyes, ugyanennyi lépésszámú megoldás is lehetséges. Az ezeknél hosszabb megoldásokra annyiszor 1 pontot kell levonni, ahány utasítással hosszabbak a mintaként megadottnál!

3. feladat: Túra (24 pont)

A. (*):  3 pont

(**):  3 pont

(***):  3 pont

B. Az ilyen pontoknál odafelé is és visszafelé is meg kell állni 3 pont

C. Pontosan: T(N)=T(N-1), azaz sík területen van a célpont 2 pont
vagy T(N)>T(N-1), azaz emelkedő végén kell megfordulni 2 pont

Kevesebbet: T(N)<T(N-1), azaz visszafordulás előtt meg kell állni 2 pont

D. A `Ciklus vége` után kell beszúrni 3 pont

`Ha T(N)<T(N-1) akkor S:=S+1` 3 pont

4. feladat: Szótagoló (25 pont)

(példák sorrendben: TEA, KASZAKŐ, ALABAMA, FAKANÁL, amire csak jó példát ad általános megfogalmazás nélkül, arra 2 pont adható.)

Első: Ha a magánhangzói között nincs mássalhangzó 3 pont
vagy 1 szótagú 2 pont

Második: Ha a magánhangzói között legfeljebb 1 mássalhangzó van 2 pont
de a mássalhangzó lehet kétjegyű (pl. sz, cs, ..) is 2 pont
vagy 1 szótagú 2 pont

Harmadik: Ha a magánhangzói között egyetlen karakter van 2 pont
a szó elején csak magánhangzó lehet 2 pont
a szó végén csak magánhangzó lehet 2 pont

Negyedik: Ha a magánhangzói között egyetlen karakter van 2 pont
vagy 1 szótagú 2 pont
a szó elején és végén legfeljebb 1 mássalhangzó van 2+2 pont

1+1 pont, ha azt írja, hogy az elején+végén nem lehet mássalhangzó, illetve akkor is, ha azt írja, hogy az elején+végén egy mássalhangzónak kell lenni.

Kilencedik-tizedik osztályosok

1. feladat: Sorszámozás (20 pont)

A. (i-1)*N+j — 4 pont

1 pont, ha a képlet csak az első sorra jó

B. (N-j)*N+i — 4 pont

1 pont, ha a képlet csak az utolsó oszlopra jó

C. (i+j-1)*(i+j-2)/2+j, ha i+j-1≤N — 4 pont

Legyen K=N-i+1, L=N-j+1! A fenti képlet folytatása:
N*N+1-(K+L-1)*(K+L-2)/2-L, ha i+j-1>N

1 pont, ha a képlet csak az első sorra jó

1 pont, ha a képlet csak az első oszlopra jó

D. max(i-1,j-1)*max(i-1,j-1)+j, ha j≤i — 4 pont

max(i-1,j-1)*max(i-1,j-1)+j+j-i, ha j>i — 4 pont

1 pont, ha a képlet csak az első sorra jó

1 pont, ha a képlet csak az első oszlopra jó

Megjegyzés: A fentiekkel egyenértékű megoldásokat is el kell fogadni!

2. feladat: Gépi nyelv (16 pont)

(az `INP A` beolvasás és az `OUT A` kiírás nélkül)

A.
```
ADD A,A
ADD A,A
ADD A,A
ADD A,A
```

B.
```
MOV A,B
ADD A,A
ADD A,B
ADD A,A
ADD A,A
ADD A,B
```

C.
```
MOV A,B
ADD A,A
ADD A,A
ADD A,A
ADD A,A
SUB A,B
ADD A,A
ADD A,B
```

D.
```
MOV A,B
ADD A,A
ADD A,A
ADD A,A
SUB A,B
ADD A,A
```

Programonként 4-4 pont. Más helyes, ugyanennyi lépésszámú megoldás is lehetséges.

Az ezeknél hosszabb megoldásokra annyiszor 1 pontot kell levonni, ahány utasítással hosszabbak a mintaként megadottnál.

3. feladat: Kocka (28 pont)

A. (0,0,0),(0,0,1),(0,1,1),(0,1,0),(0,0,0),(0,0,1),(0,1,1),(0,1,0),(0,0,0) — 7 pont

B. (0,0,0),(0,0,1),(0,1,1),(1,1,1),(1,1,0),(1,0,0),(0,0,0) — 5 pont

C. (0,0,0),(0,0,1),(0,1,1),(0,1,0),(1,1,0),(1,0,0),(1,0,1),(1,1,1),(0,1,1),(0,0,1),(0,0,0) — 9 pont

D. (0,0,0),(0,0,1),(1,0,1),(1,0,0),(1,1,0),(1,1,1),(0,1,1),(0,1,0),(0,0,0) — 7 pont

Az első két pozícióra nem jár pont. Attól kezdve annyi pont jár, ahány pozíció folyamatosan helyesen szerepel.

4. feladat: Szótagoló (18 pont)

(Példák sorrendben: KICSIKE, HÁTRÁLTAT, SAVANYÚ, TEA, amire csak jó példát ad általános megfogalmazás nélkül, arra 2 pont adható.)

Első: Mássalhangzóval kezdődő szavak	1 pont
magánhangzói között egyetlen (akár kétjegyű) mássalhangzó van	1 pont
magánhangzóra végződik	1 pont
vagy az utolsó két magánhangzója között nincs mássalhangzó	1 pont
vagy 1 szótagú magánhangzóval kezdődő vagy végződő	2 pont
Második: Legfeljebb egy mássalhangzóval kezdődő szavak	1 pont
legfeljebb 1 mássalhangzóra végződik	1 pont
magánhangzói között két (egyjegyű) mássalhangzó van	1 pont
vagy 1 szótagú legfeljebb 1 mássalhangzóval kezdődő és végződő	1 pont
Harmadik: Bármivel kezdődő szavak	1 pont
magánhangzói között egyetlen (akár kétjegyű) mássalhangzó van	2 pont
magánhangzóra végződik	1 pont
vagy 1 szótagú, magánhangzóra végződő	1 pont
Negyedik: Ha a magánhangzói között nincs mássalhangzó	2 pont
vagy 1 szótagú	1 pont

5. feladat: Rácsháló (18 pont)

A. 8 tartomány 3 pont

B. 15 tartomány 3 pont

C. 8 tartomány 3 pont

D. **9** tartomány 3 pont

E. 6 tartomány 3 pont

F. 11 tartomány 3 pont

Mindegyik esetben 1 pont adható, ha a minimálisnál eggyel nagyobb lépésszámot ad.

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Sorszámozás (21 pont)

A. (i-1)*N+j 3 pont

1 pont, ha a képlet csak az első sorra jó

B. (j-1)*N+i, ha **j** páratlan 3 pont

(j-1)*N+N-i+1, ha **j** páros 3 pont

1 pont, ha a képlet csak az első oszlopra jó

C. max(i-1,j-1)*max(i-1,j-1)+2*j-1, ha j≤i 3 pont

max(i-1,j-1)*max(i-1,j-1)+2***i**, ha j>i 3 pont

1 pont, ha a képlet csak az első sorra, vagy az első oszlopra vagy a főátlóra jó

D. max(i-1,j-1)*max(i-1,j-1)+j, ha j≤i 3 pont

max(i-1,j-1)*max(i-1,j-1)+j+j-i, ha j>i 3 pont

1 pont, ha a képlet csak az első sorra jó vagy az első oszlopra jó

2. feladat: Vonatok (16 pont)

A. Az egyes vonatok menetidőit 3 pont

B. Az egyik állomásról elindult, de a másikra még meg nem érkezett vonatok indulási idői vannak benne 3 pont

C. Annak az állomásnak a sorszámát tárolja, amelyről a legutóbbi vonat indult 3 pont

D. (), (1), (1,3)**, (3);** (3,4), (3,4,5)**;** (4,5), (5)**;** (), (8)**;** (8,9), (9)**;** 5 pont
ha a **;**-vel jelölt határokig jó, akkor annyi pont, amennyi pontosvesszőig eljutott.

E. Legkisebb: 2 1 pont
Legnagyobb: 5 1 pont

3. feladat: Hálózat (24 pont)

A. 8 pont	B: 8 pont	C. 8 pont
1: NINCS	1: NINCS	1: NINCS
2: NINCS	2: NINCS	2: NINCS
3: NINCS	3: NINCS	3: NINCS
4: (1-2,2-3,3-4,4-5),400	4: (1-3,1-4,1-5,2-3),460	4: (1-5,2-3,3-4,4-5),460
5: (1-2,1-3,3-4,4-5),380	5: (1-4,1-5,2-3,3-5),450	5: (1-2,1-5,2-3,4-5),430
6: (1-2,1-3,3-4,3-5),360	6: (1-4,1-5,2-4,3-5),440	6: (1-2,2-3,2-4,4-5),410
7: (1-2,1-3,2-4,3-5),340	7: (1-4,1-5,2-4,3-4),410	7: (1-2,2-4,3-5,4-5),400
8: (1-3,2-4,2-5,3-5),320	8: (1-4,1-5,2-5,3-4),390	8: (1-2,2-4,3-5,4-5),380
	9: (1-4,1-5,2-5,3-4),390	9: (1-2,2-4,3-4,4-5),360
	10: (1-5,2-5,3-4,4-5),360	10: (1-2,1-3,3-4,4-5),340

Minden helyes lépésre 1 pont adható, a B és a C feladatban az első 2 lépésre nem jár pont. Ha csak a párok jók, vagy csak a költség jó, akkor a pontszám fele adható, a végén lefelé kerekítve.

4. feladat: Mobiltelefon (20 pont)

(más, az alábbiakkal azonos lépésszámú, helyes megoldások is elfogadhatók, az egyes lépések sorrendje pedig tetszőleges lehet)

A. XOR(2,2,8,8), XOR(3,5,7,5), XOR(5,3,5,7), XOR(5,5,5,5) 4 pont

B. XOR(2,2,8,8), XOR(5,3,8,5), XOR(7,3,7,7) 4 pont

C. XOR(4,2,6,8), XOR(5,3,8,5), XOR(2,5,5,7) 4 pont

D. XOR(1,1,9,9), XOR(1,4,9,6), XOR(4,1,6,9) 4 pont

E. XOR(1,**3**,3,9), XOR(6,1,7,6), XOR(3,2,8,3), XOR(2,6,8,8) 4 pont

Nagyobb lépésszámú megoldás esetén ebből annyi pontot kell levonni, ahány lépéssel hosszabb (de 0-nál kevesebb pont nem adható).

Javítási tanács: ha egy rácshálóban egy pontot teszünk az egyes XOR műveleteknek megfelelő téglalapok összes mezőjébe, akkor a végén azokat a mezőket kell besatírozni, amelyekben páratlan számú pont van.

5. feladat: Véletlen (19 pont)

Az egyes részkérdések magyarázatára az alábbiakkal egyenértékű megoldások is elfogadhatók, azaz a lényeg, hogy a versenyző rájöjjön a problémára.

A. Nem helyes 2 pont

X(N) biztosan marad, X(N-1) (K-1)/K eséllyel marad, ... 2 pont
(azaz a növekvő sorszámúak egyre nagyobb eséllyel kerülnek kiválasztásra)

N=K+1-re sem helyes 1 pont

B. Nem helyes 2 pont

X(N) K/(K+1) eséllyel marad, X(N-1) ugyanekkora eséllyel kerül Y-ba, de 1/(K+1) eséllyel a helyére kerülhet, azaz ennél kisebb eséllyel marad, ... 2 pont
(azaz a növekvő sorszámúak egyre nagyobb eséllyel kerülnek kiválasztásra)

N=K+1-re helyes 1 pont

C. Helyes 2 pont

D. Nem helyes 2 pont

Bár mindegyik X-beli K/N eséllyel kerül Y-ba, nem garantált, hogy a végén pontosan K darab elem lesz Y-ban (lehet több is, kevesebb is) 2 pont

N=K+1-re sem helyes 1 pont

E. Helyes 2 pont

2004. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Gyufák (23 pont)

Négy gyufára nincs megoldás, az össze többi esetre pedig olyan számhármasokat kell megvizsgálni (a>b>c esetre), amelyek lehetnek egy háromszög oldalai.

```
Gyufák(n):
  Ha n=4 akkor Ki: 0
  különben db:=0
           Ciklus a=1-től n div 3-ig
             Ciklus b=a-tól (n-a) div 2-ig
               c:=n-a-b
               Ha a+b>c akkor db:=db+1; Ki: a,b,c)
             Ciklus vége
           Ciklus vége
           Ki: db
Eljárás vége.
```

2. feladat: Legolcsóbbak (24 pont)

Ha még nem volt K nap, akkor sorbarendezve gyűjtjük az árakat és a napsorszámokat (`T(i).ár`, illetve `t(i).sorszám`). Ha már volt K nap, akkor ha a K-adik legolcsóbbnál olcsóbb az alma, akkor beillesztjük a tároltak közé.

```
Almák(n,k):
  h:={}
  Ciklus i=1-től n-ig
    Be: ár
    Ha i≤k akkor j:=i-1
                Ciklus amíg j>0 és ár<t(j).ár
                  t(j+1):=t(j); j:=j-1
                Ciklus vége
                t(j+1).ár:=ár; t(j+1).sorszám:=i; h:=h+{i}
    különben ha ár<t(k).ár
           akkor j:=k-1; h:=h-{t(k).sorszám}
                Ciklus amíg j>0 és ár<t(j).ár
                  t(j+1):=t(j); j:=j-1
                Ciklus vége
                t(j+1).ár:=ár; t(j+1).sorszám:=i; h:=h+{i}
    Ha i≥k akkor Ciklus j=1-től i-ig
                  Ha j∈h akkor Ki: j
                Ciklus vége
  Ciklus vége
Eljárás vége.
```

3. feladat: Tanév (28 pont)

Az első tanítási nap (`tk`) szeptember első hétfője, azaz szeptember 1, ha az hétfőre esik, különben pedig az őt követő első hétfő.

Az őszi szünet legkésőbb október 21-én kezdődik (`oe`), ha október 23. hétfőre esne, különben az őt megelőző szombat. Ehhez felhasználhatjuk, hogy október 23. az 54. napja a tanévnek. Az őszi szünet utolsó napja (`ou`) is biztosan októberben van, 10 nappal a kezdete után.

A téli szünet legutolsó lehetséges kezdete december 22, a karácsony a tanév 116. napja, ezekből a téli szünet pontos kezdete (`tle`) számítható. Az utolsó nap (`tlu`) az ezt követő 16. nap, de ez már biztosan a következő év januárjára esik.

A tavaszi szünetnél a hónap kétféle lehet, március vagy április, a húsvéttól függően. A kezdete a húsvét vasárnap előtti 9. nap (`tveh,tve`), a vége pedig 2 nappal lesz húsvét vasárnap után (`tvuh,tvu`).

A tanév vége biztosan júniusra esik, legkésőbb június 14-re (`tv`).

```
Tanév(év,szept1,husho,husnap):
  Ha szept1=1 akkor tk:=1 különben tk:=9-szept1
  oe:=21-(54-tk) mod 7; ou:=oe+10
  tle:=22-(116-tk) mod 7; tlu:=16-(30-tle)
  tve:=husnap-9; tveh:=4
  Ha tve≤0 akkor tve:=tve+31; tveh:=3
  tvu:=husnap+2; tvuh:=husho
  Ha tvu>31 akkor tvu:=tvu-31; tvuh:=4
  Ha husho=3 akkor husnap:=husnap-5 különben husnap:=husnap-2
  tv:=14-(61-husnap) mod 7
  Ki: 'A tanév kezdete:',ev,'.9.',tk,'.'
  Ki: 'Az őszi szünet előtti utolsó tanítási nap:',
      ev,'.10.',oe,'.'
  Ki: 'Az őszi szünet utáni első tanítási nap:',
      ev,'.10.',ou,'.'
  Ki: 'A téli szünet előtti utolsó tanítási nap:',
      ev,'.12.',tle,'.'
  Ki: 'A téli szünet utáni első tanítási nap:',
      ev+1,'.1.',tlu,'.'
  Ki: 'A tavaszi szünet előtti utolsó tanítási nap:',
      ev+1,'.',tveh,'.',tve,'.')
  Ki: 'A tavaszi szünet utáni első tanítási nap:',
      ev+1,'.',tvuh,'.',tvu,'.')
  Ki: 'A tanév vége:',ev+1,'.6.',tv,'.'
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Másolatok (15 pont)

A feladat kitűzésének évében az összes szöveg nem fért el a memóriában, így ügyes tárolást kellett választani. Természetesen nagyobb paraméterkorlátokkal ez a probléma ma is fennáll.

Tároljuk a `hszam` elemű `hibák` tömbben a felfedezett eltéréseket! Ha a szöveg valamely pozíciójában eltérést fedezünk fel az eredeti szövegtől (`eredeti`), akkor az eredeti szöveg megfelelő helyére tegyünk #-karaktert, majd a végén – ha lehetséges – cseréljük a többségi karakterre!

Ha egy pozíción már volt eltérés, akkor meg kell nézni, hogy az aktuális karakter szerepelt-e már ezen a pozíción (`hibák(ii).c(jj)`)! Ha igen, akkor növelni kell az előfordulás számát (`hibák(ii).d(jj)`), különben pedig fel kel venni a `hibák` tömbbe az új karaktert is.

Ha most van először eltérés az aktuális pozíción, akkor fel kell venni a `hibák` tömbbe az eredeti karaktert is és az új karaktert is!

```
Másolatok(n,m,eredeti):
  hszam:=0
  Ciklus i=2-től n-ig
    Ciklus j=1-től m-ig
      Olvas(BeF,szöveg(j))
      Ha eredeti(j)='#'
         akkor ii:=1
               Ciklus amíg hibák(ii).poz≠j
                 ii:=ii+1
               Ciklus vége
               jj:=1
               Ciklus amíg jj≤hibák(ii).db és
                           hibák(ii).c(jj)≠szoveg(j)
                 jj:=jj+1
               Ciklus vége
               Ha jj≤hibák(ii).db
                  akkor hibák(ii).d(jj):=hibák(ii).d(jj)+1
               különben hibák(ii).db:=hibák(ii).db+1
                        hibák(ii).c(db):=szoveg(j)
                        hibák(ii).d(db):=1
      különben ha szoveg(j)≠eredeti(j)
         akkor hszam:=hszam+1
               hibák(hszam).db:=2; hibák(hszam).poz:=j
               hibák(hszam).c(1):=eredeti(j)
               hibák(hszam).d(1):=i-1
               hibák(hszam).c(2):=szoveg(j)
               hibák(hszam).d(2):=1; eredeti(j):='#'
    Ciklus vége
  Ciklus vége
  Ciklus i=1-től m-ig
    Ha eredeti(i)='#' akkor eredeti(i):=Cserélt(i)
  Ciklus vége
Eljárás vége.
```

Az i-edik pozíción biztosan van hiba, ezért meg kell keresni a `hibák` tömbben a helyét! Ezután le kell számolni, hogy ezen a pozíción összesen hány karaktervváltozat volt, s ha van olyan változat, ami az esetek több mint felében előfordul, akkor azt kell venni helyes karakternek.

```
Cserélt(i):
  j:=1
  Ciklus amíg i≠hibák(j).poz
    j:=j+1
  Ciklus vége
  s:=0
  Ciklus k=1-től hibák(j).db
    s:=s+hibák(j).d(k)
  Ciklus vége
  k:=1
  Ciklus amíg k≤hibák(j).db és hibák(j).d(k)≤s/2.0
    k:=k+1
  Ciklus vége
  Ha k≤hibák(j).db akkor Cserélt:=hibák(j).c(k)
                  különben Cserélt:='#'
Függvény vége.
```

2. feladat: Alma (12 pont)

Naponta figyeljük a k legolcsóbb árust. Az input file-ban előreolvasunk egy rekordot, a következő árus érkezését. Azon a napon van teendőnk, amely napon jött árus (több is lehet egymás után).

Ekkor meg kell néznünk, hogy árult-e már korábban! Ha szerepelt, akkor el kell távolítani: ki kell szedni a legolcsóbb k halmazából (h), valamint az árak szerint rendezett tömbből is ki kell hagyni! Ha már több, mint k árusunk van, akkor a k+1-ediket kell bevennünk a legolcsóbbak közé!

Ezután az új árat be kell szúrnunk a rendezett tömbbe, s ha a legolcsóbb k közé kerül, akkor a h halmazba is fel kell venni! Ha a halmazban már volt k elem, akkor a legnagyobbat ki kell hagyni.

Minden nap végén meg kell nézni, hogy van-e k árus, s ha igen, akkor a h halmaz elemeit kell kiírni!

```
Almák(n,m,k):
  Olvas(BeF,nap,sorszám,ár); db:=0; h:={}
  Ciklus ns=1-től n-ig
    Ciklus amíg nap=ns
      j:=1
      Ciklus amíg j≤db és t(j).sorszám≠sorszám
        j:=j+1
      Ciklus vége
      Ha j≤db akkor
        Ha j≤k akkor h:=h-{t(j).sorszám}
                     Ha db>k akkor h:=h+{t(k+1).sorszám}
        Ciklus amíg j≤db
          t(j):=t(j+1); j:=j+1
        Ciklus vége
        db:=db-1
      Elágazás vége
      j:=db
      Ciklus amíg j>0 és ár<t(j).ár
        t(j+1):=t(j); j:=j-1
      Ciklus vége
      t(j+1).ár:=ár; t(j+1).sorszám:=sorszám; db:=db+1
      Ha j+1≤k akkor h:=h+{sorszám}
                     Ha db>k akkor h:=h-{t(k+1).sorszám}
      Ha nem vége?(BeF) akkor Olvas(BeF,nap,sorszám,ár)
                     különben nap:=n+1
    Ciklus vége
    Ha db≥k akkor Ciklus i=1-től m-ig
                    Ha i∈h akkor Ír(KiF,i)
                  Ciklus vége
          különben Ír(KiF,0)
  Ciklus vége
Eljárás vége.
```

3. feladat: Pakolás (15 pont)

Minden pozícióra számoljuk ki, hogy hány B (NB), hány C (NC) és hány BC (B-ben C) láda (NBC) van tőle balra, amit még lehet pakolni! Számoljuk azt is, hogy hány ládát mozgattunk már! A feladat mozgatása a ládák számából (N) kivonva a mozgatott ládák számát (Pakol).

Ha A típusú láda jön, akkor ha B és C típusú is van tőle balra, akkor mindkettő belerakható. Ha BC típusú van tőle balra, az is belerakható. Ha csak B vagy csak C típusú van, az is belerakható.

Ha B típusú láda jön, akkor ha van tőle balra C típusú, akkor belerakható, s keletkezik egy mozgatható BC típusú, ha pedig nincs, akkor keletkezik egy mozgatható B típusú.

Ha C típusú láda jön, akkor keletkezik egy mozgatható C típusú.

```
Pakolás(N,Pakol):
  Pakol:=0; NB:=0; NC:=0; NBC:=0
  Ciklus i=1-től N-ig
    Olvas(BeF,X)
    Ha X='A'
       akkor Ha NB>0 és NC>0
                akkor NB:=NB-1; NC:=NC-1; Pakol:=Pakol+2
             különben ha NBC>0
                akkor NBC:=NBC-1; Pakol:=Pakol+1
             különben ha NB>0
                akkor NB:=NB-1; Pakol:=Pakol+1
             különben ha NC>0
                akkor NC:=NC-1; Pakol:=Pakol+1
    különben ha X='B'
       akkor Ha NC>0
                akkor NC:=NC-1; Pakol:=Pakol+1; NBC:=NBC+1
             különben NB:=NB+1
    különben ha X='C' akkor NC:=NC+1
  Ciklus vége
Eljárás vége.
```

```
  WriteLn(KiF,N-Pakol);
```

4. feladat: Kivágás (15 pont)

A feladat szerint `M` pontról kell eldönteni, hogy egy vízszintes és függőleges szakaszokkal határolt zárt területen belül vannak-e vagy sem.

Egyszerű esetben azt kellene megszámolni, hogy az (`x0,y0`) pontból balra húzott vízszintes vonal páratlan számú függőleges szakaszt metsz-e. Ez mindenesetben jól működik, ha az ilyen vízszintes szakasz nem megy át az alakzat valamely vízszintes szakaszán.

A vízszintes szakaszokon áthaladás esetén azt kell nézni, hogy a szakasz két végén a függőleges szakasz mely irányba folytatódik.

```
Kivágás(N,P,x0,y0,Belül):
  Metsz:=0
  Ciklus i=1-től N-ig
    ii:=i+1; Ha ii>N akkor ii:=1
    Ha P(i).y=P(ii).y és P(i).y<y0 akkor
      Ha P(i).x<P(ii).x és P(i).x<x0 és x0<P(ii).x vagy
         P(i).x>P(ii).x és P(ii).x<x0 és x0<P(i).x
            akkor Metsz:=Metsz+1
    Ha P(i).x=P(ii).x és x0=P(i).x akkor
      Ha P(i).y<P(ii).y akkor yy:=P(ii).y
                    különben yy:=P(i).y
      Ha yy<y0 akkor
        Ha i=1 akkor i1:=N különben i1:=i-1
        iii:=i Mod N+2
        Ha P(i1).x<x0 és x0<P(iii).x vagy
           P(iii).x<x0 és x0<P(i1).x akkor Metsz:=Metsz+1
  Ciklus vége
  Belül:=Páratlan(Metsz)
Eljárás vége.
```

5. feladat: Rács (18 pont)

A feladat arról szól, hogy hogyan lehet ügyesen kódolni az üvegrács nyolc- és négyszög alakú mezőit. Az ábra alapján látszik, hogy minden négyzet egy nyolcszögtől északkeletre van, azaz (`i,j,1`) legyen a nyolcszög, (`i,j,2`) pedig a tőle északketeletre levő négyzet pozíciója! Jelölje `t(i,j,k)`

azt, hogy hányszor jártunk az (i,j,k) pozíción! (kx,ky,kh) legyen az aktuális pozíció, dba az érintett négyzetek, dbc pedig a többször érintett mezők száma!

```
Rács(kx,ky,dba,dbc)
  t():=0; kh:=1; t(kx,ky,kh):=1; dba:=0; dbc:=0
  Ciklus amíg nem vége?(BeF)
    Olvas(BeF,c)
    Ha kh=1 akkor Ha c='E' akkor ky:=ky+1
                  különben ha c='K' akkor kx:=kx+1
                  különben ha c='D' akkor ky:=ky-1
                  különben ha c='N' akkor kx:=kx-1
                  különben ha c='EK' akkor kh:=2
                  különben ha c='DK' akkor kh:=2; ky:=ky-1
                  különben ha c='DN' akkor kh:=2; kx:=kx-1
                                           ky:=ky-1
                  különben ha c='EN' akkor kh:=2; kx:=kx-1
    különben Ha c='EK' akkor kh:=1; kx:=kx+1; ky:=ky+1
             különben ha c='DK' akkor kh:=1; kx:=kx+1
             különben ha c='DN' akkor kh:=1
             különben ha c='EN' akkor kh:=1; ky:=ky+1
    Ha kh=2 és t(kx,ky,kh)=0 akkor dba:=dba+1
    Ha t(kx,ky,kh)=1 akkor dbc:=dbc+1
    t(kx,ky,kh):=t(kx,ky,kh)+1
  Ciklus vége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Őr (15 pont)

Minden őrnél tároljuk, hogy a haladási iránya függőleges-e (hely(i).irány), a lépése irányát (hely(i).lép): +1, ha növekvő, -1, ha csökkenő indexű helyre fog lépni. Tudni kell továbbá minden őr kezdő sor- és oszlopindexét (hely(i).ks, hely(i).ko), valamint az aktuális helyét (hely(i).s,hely(i).o). Ismerjük még azt is, hogy az őrök mennyire távolodhatnak el a kiinduló helyükről (hely(i).l).

Könnyen kiszámíthatjuk, hogy minden első találkozáshoz legfeljebb a legnagyobb elmozdulási távolság négyzete időn belül sor kerülhet.

Az előkészítés után tehát követni kell az őrök mozgását, ha a maximális lépéstávolságra értek, akkor a haladási irányukat meg kell változtatni, majd figyelni kell, hogy találkoztak-e!

```
Őr(n,m,k,db,talál):
  ml:=0; tdb:=0
  Ciklus i=1-től k-ig
    Olvas(BeF,c); hely(i).irány:=(c='F')
    Ha hely(i).irány akkor hely(i).lép:=1
                  különben hely(i).lép:=-1
    Olvas(BeF,hely(i).ks,hely(i).ko,hely(i).l)
    hely(i).s:=hely(i).ks; hely(i).o:=hely(i).ko
    Ha hely(i).l>ml akkor ml:=hely(i).l
  Ciklus vége
  Találkozásvizsgálat(0,talál,tdb)
```

```
  Ciklus i=1-től ml*ml-ig
    Ciklus j=1-től k-ig
      Ha hely(j).irány akkor
        Ha |hely(j).s-hely(j).ks|=1
           akkor hely(j).lép:=-hely(j).lép
        hely(j).s:=hely(j).s+hely(j).lép
      különben ha |hely(j).o-hely(j).ko|=1
                  akkor hely(j).lép:=-hely(j).lép
               hely(j).o:=hely(j).o+hely(j).lép
    Cikklus vége
    Találkozásvizsgálat(i,talál,tdb)
  Ciklus vége
Eljárás vége.
```

Két őr először találkozik a `mikor` időpontban, ha azonos a sor- és az oszlop-koordinátájuk, és eddig még nem találkoztak.

```
Találkozásvizsgálat(mikor,talál,tdb):
  Ciklus i=1-től k-1-ig
    Cilus j=i+1-től k-ig
      Ha hely(i).s=hely(j).s és hely(i).o=hely(j).o
                              és nem Volt(i,j,talál,tdb)
         akkor tdb:=tdb+1
               talál(tdb).or1:=i; talál(tdb).or2:=j
               talál(tdb).s:=hely(i).s
               talál(tdb).o:=hely(i).o
               talál(tdb).ido:=mikor
    Ciklus vége
  Ciklus vége
Eljárás vége.

Volt(i,j,talál,tdb):
  k:=1
  Ciklus amíg k≤tdb és (talál(k).or1≠i vagy talál(k).or2≠j)
    k:=k+1
  Ciklus vége
  Volt:=(k≤tdb)
Függvény vége.
```

2. feladat: Szövegek (12 pont)

Csak akkor érdemes a másolatokkal foglalkozni, ha van legalább 1 másolat, azaz `n>1`. A második másolat biztos az elsőből keletkezett. Csupán annyit kell tenni, hogy feljegyezzük azon pozíciókat, amelyeken a második szöveg eltér az elsőtől.

A többieknél meg kell keresni, hogy melyiktől a legkevesebb az eltérése.

```
Másol(n,szöveg):
  Ír(Kif,1); j:=hossz(szöveg(1))
  másolt(2).db:=0; {A második eltéréshalmaza az elsőtől}
  Ciklus i=1-től j-ig
    Ha szöveg(1,i)≠szöveg(2,i)
       akkor másolt(2).db:=másolt(2).db+1
             másolt(2).t(másolt(2).db):=i
  Ciklus vége
```

```
  Ciklus c=3-tól n-ig
    másolt(c).db:=0; min:=1     {A c-edik eltéréshalmaza}
    Ciklus i=1-től j-ig
      Ha szöveg(1,i)≠szöveg(c,i)
       akkor másolt(c).db:=másolt(c).db+1
             másolt(c).t(másolt(c).db):=i
    Ciklus vége
    Ciklus d=2-től c-1-ig   {A c d-től vett eltéréshalmaza}
      Ha Másolható(d,c) és másolt(1).db<másolt(c).db
         akkor másolt(c):=másolt(1); min:=d
    Ciklus vége
    Ír(KiF,min)
  Ciklus vége
Eljárás vége.
```

Mivel feltehettük, hogy egy betű kétszer nem változott, ezért azt kell néznünk `c` betűiről, hogy megegyeznek-e `d` megfelelő betűjével, vagy pedig az adott pozíción `d` még nem változott!

```
Másolható(d,c):
  i:=1; másolt(1).db:=0
  Ciklus amíg i≤j és (szöveg(d,i)=szöveg(c,i))
                      vagy Nincsbenne(i,d))
    Ha szöveg(d,i)≠szöveg(c,i)
       akkor másolt(1).db:=másolt(1).db+1
             másolt(1).t(másolt(1).db):=i
    i:=i+1
  Ciklus vége
  Másolható:=(i>j)
Függvény vége.
```

Megnézzük, hogy az i-edik pozíció szerepel-e a `d` eltéréshalmazában.

```
Nincsbenne(i,d):
  j:=1
  Ciklus amíg j≤másolt(d).db és i≠másolt(d).t(j))
    j:=j+1
  Ciklus vége
  Nincsbenne:=(j>másolt(d).db)
Függvény vége.
```

3. feladat: Hálózati központ (15 pont)

A feladat szerint tudjuk, hogy a hálózat egy fával írható le. Az eredmény kiszámításához hagyjuk el a fa összes levelét, amíg 1-2 pont nem marad! Ha egy maradt, akkor ő a központ, ha pedig kettő, akkor bármelyikük lehet az.

Beolvasáskor számítsuk ki minden pont szomszédai számát (`fok`), valamint a fát az éleivel ábrázoljuk (`Él`)! Ekkor természetsen tudnunk kell, hogy egy pont első éle hol található az élek tömbjében (`ElsőÉl`).

```
Beolvas:
  Ciklus u=1-től N-ig
    Fok(u):=0; ElsőÉl(u):=Eszam+1; Olvas(BeF,v)
    Ciklus amíg v>0
      Fok(u):=Fok(u)+1; Eszam:=Eszam+1; Él(Eszam):=v
      Olvas(BeF,v)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A levelek első lépésként az 1 szomszédszámú pontok. Ha őket elhagyjuk, akkor újabb 1 szomszédszámú pontok keletkeznek. Ezeket kell a második menetben elhagyni. Azaz egyszerre kétféle pontokat kell kezelnünk: a most már 1 szomszédúakat (Régi) és a most 1 szomszédúvá válókat (Új). Mindkettőt tehetjük a Fok1 halmazba, melyek elemszáma LevRégi, illetve LevÚj..

```
Számít:
  Régi:=igaz; Új:=hamis; LevÚj:=0; LevRégi:=0
  Ciklus i=1-től N-ig
    Van(i):=igaz
    Ha Fok(i)=1 akkor LevRégi:=LevRégi+1 {1 fokuak halmaza}
                      Fok1(Régi,LevRégi):=i
  Ciklus vége
  Maradt:=N
  Ciklus amíg Maradt>2
    LevÚj:=0
    Ciklus i=1-től LevRégi-ig
      u:=Fok1(Régi,i); Van(u):=hamis
      Ciklus ii:=ElsőÉl(u)-tól ElsőÉl(u+1)-1-ig
        v:=Él(ii)    {az u pont szomszédai}
        Ha Van(v)
           akkor Fok(v):=Fok(v)-1
                 Ha Fok(v)=1 akkor LevÚj:=LevÚj+1
                                   Fok1(Új,LevÚj):=v
      Ciklus vége
    Ciklus vége
    Régi:=Új; Új:=Nem Új
    Maradt:=Maradt-LevRégi; LevRégi:=LevÚj
  Ciklus vége
  Mego1:=Fok1(Régi,1)
  Ha LevRégi=2 akkor Mego2:=Fok1(Régi,2)
             különben Mego2:=0
Eljárás vége.
```

4. feladat: Fényképezés (15 pont)

A feladat a legkevesebb elemszámú olyan halmaz (H) meghatározása, hogy minden intervallumba esik a halmaznak legalább 1 pontja.

Mivel az időpontok száma kicsi (1000), ezért beolvasáskor tároljuk minden intervallum végre, azaz távozási időre (t), hogy hol van a kezdete (Int(t))! Ha több azonos távozási idő is van, akkor közülük elég a legkésőbb érkezőt tárolni.

A mohó megoldásban tároljuk az utolsó hamazba választott pontot, majd ha az aktuális intervallum később kezdődik, akkor az ő végében kell fényképezni. Belátható ugyanis, hogy a leghamarabb elmenő távozása előtti utolsó pillanatban kell először fényképezni, majd a maradékból a leghamarabb olyan elmenőt, aki az első elmenő után jött, …

```
Fénykép(MaxT,Int):
  Utolsó:=0       {az utolsó beválasztott pont}
  M:=0            {a beválasztott pontok száma}
  Ciklus x=1-től MaxT-ig
   Ha Int(x)>0 és Utolsó<Int(x)
      akkor Utolsó:=x-1; M:=M+1; Mego(M):=Utolsó
  Ciklus vége
Eljárás vége.
```

5. feladat: Rácsháló (18 pont)

A feladat arról szól, hogy hogyan lehet ügyesen kódolni az üvegrács tizenkét- (KH=1), négy- (KH=4) és háromszög (KH=2,3,5,6) alakú mezőit. Az ábra alapján látszik, hogy tizenkétszögtől

északkeletre 4 háromszög és 1 négyzet van. Azaz (i,j,1) legyen a tizenkétszög, (i,j,2..6) pedig a tőle északketeletre levő háromszögek, illetve négyzet pozíciója! Jelölje t(i,j,k) azt, hogy hányszor jártunk az (i,j,k) pozíción! (kx,ky,kh) legyen az aktuális pozíció, dba az érintett négyzetek, dbc pedig a többször érintett mezők száma!

```
Rács(kx,ky):
  t():=0; kh:=1; t(kx,ky,kh):=1; dba:=0; dbc:=0
  Ciklus amíg nem vége(f)
    Olvas(BeF,c)
    Ha kh=1 akkor              {ez a tizenkétszög}
       Ha c='E' akkor inc(ky)
       különben ha c='K' akkor inc(kx)
       különben ha c='D' akkor dec(ky)
       különben ha c='N' akkor dec(kx)
       különben ha c='EEK' akkor kh:=2
       különben ha c='KEK' akkor kh:=3
       különben ha c='KDK' akkor kh:=5; dec(ky)
       különben ha c='DDK' akkor kh:=2; dec(ky)
       különben ha c='DDN' akkor kh:=6; dec(kx); dec(ky)
       különben ha c='NDN' akkor kh:=5; dec(kx); dec(ky)
       különben ha c='NEN' akkor begin kh:=3; dec(kx)
       különben ha c='EEN' akkor begin kh:=6; dec(kx)
    különben ha kh=2 akkor
       Ha c='EEN' akkor kh:=1; inc(ky)
       különben ha c='DDN' akkor kh:=1
       különben ha c='K' akkor kh:=4
    különben ha kh=3 akkor
       Ha c='NDN' akkor kh:=1
       különben ha c='KDK' akkor kh:=1; inc(kx)
       különben ha c='E' akkor kh:=4
    különben ha kh=4 akkor    {ez a négyzet}
       Ha c='E' akkor kh:=5
       különben ha c='K' akkor kh:=6
       különben ha c='D' akkor kh:=3
       különben ha c='N' akkor kh:=2
    különben ha kh=5 akkor
       Ha c='NEN' akkor kh:=1; inc(ky)
       különben ha c='KEK' akkor kh:=1; inc(kx); inc(ky)
       különben ha c='D' akkor kh:=4
    különben ha kh=6 akkor
       Ha c='EEK' akkor kh:=1; inc(kx); inc(ky)
       különben ha c='DDK' akkor kh:=1; inc(kx)
       különben ha c='N' akkor kh:=4
    Ha kh=4 és t(kx,ky,kh)=0 akkor dba:=dba+1
    Ha t(kx,ky,kh)=1 akkor dbc:=dbc+1
    t(kx,ky,kh):=t(kx,ky,kh)+1
  Ciklus vége
Eljárás vége.
```

2004. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Automata (27 pont)

Próbáljuk ki először, hogy egyetlen pénzérmével fizethetünk-e, majd kettővel, s végül hárommal!

```
Automata(ár,n,e):
  i:=1
  Ciklus amíg i≤n és e(i)≠ár
    i:=i+1
  Ciklus vége
  Ha i≤n akkor Ki: ár,'=',e(i)
  különben i:=0
           Ciklus
             i:=i+1; j:=i
             Ciklus
               j:=j+1
             amíg j≤n és e(i)+e(j)≠ár
             Ciklus vége
           amíg i≤n-1 és e(i)+e(j)≠ár
           Ciklus vége
           Ha i≤n-1 akkor Ki: ár,'=',e(i),'+',e(j)
           különben i:=0
                    Ciklus
                      i:=i+1; j:=i
                      Ciklus
                        j:=j+1; k:=j
                        Ciklus
                          k:=k+1
                        amíg k≤n és e(i)+e(j)+e(k)≠ár
                        Ciklus vége
                      amíg j≤n-1 és (e(i)+e(j)+e(k)≠ár
                      Ciklus vége
                    amíg i≤n-2 és (e(i)+e(j)+e(k)≠ár
                    Ciklus vége
                    Ha i≤n-2 akkor
                          Ki: ár,'=',e(i),'+',e(j),'+',e(k)
                          különben Ki: 'NEM LEHET'
Eljárás vége.
```

2. feladat: Szólánc (22 pont)

Mivel csak egyetlen jó sorrend lehet, ezért tároljuk minden betűhöz, hogy melyik szó első (k), illetve utolsó (v) betűje! Ha van olyan betű, ami csak kezdőbetűként fordul elő, akkor a szóláncot vele kezdjük, ha nincs ilyen akkor bármelyikkel, azaz az első szóval is kezdhetjük.

```
Szólánc(n,szó):
  k:=(0,...,0); v:=(0,...,0)
  Ciklus i=1-től n-ig
    k(szó(i,1)):=i; v(szó(i,hossz(szó(i)))):=i
  Ciklus vége
```

```
  c:='A'
  Ciklus amíg c<'Z' és nem (k(c)>0 és v(c)=0)
    c:=következő(c)
  Ciklus vége
  Ha c<'Z' akkor i:=k(c) különben i:=1
  Ki: s(i)
  Ciklus db=2-től n-ig
    j:=1
    Ciklus amíg szó(i,hossz(szó(i)))≠szó(j,1)
      j:=j+1
    Ciklus vége
    Ki: s(j); i:=j
  Ciklus vége
Eljárás vége.
```

3. feladat: Idő (26 pont)

Az idő egy négyjegyű szám, aminek az első számjegyét 24-es, a következő kettőt 60-as, az utolsót pedig 100-as számrendszerben adjuk meg. Ilyen számrendszerű számokra kell írni összeadás és kivonás műveletet!

```
Idő(o,p,mp,szmp):
  h:=szmp-d; at:=0
  Ciklus amíg h<0
    h:=h+100; at:=at+1
  Ciklus vége
  g:=mp-at-c; at:=0
  Ciklus amíg g<0
    g:=g+60; at:=at+1
  Ciklus vége
  f:=p-at-b; at:=0
  Ciklus amíg f<0
    f:=f+60; at:=at+1
  Ciklus vége
  e:=o-at-a; at:=0
  Ciklus amíg e<0
    e:=e+24; at:=at+1
  Ciklus vége
  Ki: e,f,g,h
  h:=szmp+d; at:=0
  Ciklus amíg h>99
    h:=h-100; at:=at+1
  Ciklus vége
  g:=mp+at+c; at:=0
  Ciklus amíg g>59
    g:=g-60; at:=at+1
  Ciklus vége
  f:=p+at+b; at:=0
  Ciklus amíg f>59
    f:=f-60; at:=at+1
  Ciklus vége
  e:=o+at+a; at:=0
  Ciklus amíg e>23
    e:=e-24; at:=at+1
  Ciklus vége
  Ki: e,f,g,h
Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Rémhír (15 pont)

Az emberek kapcsolatait egy gráffal írhatjuk le. Az A részfeladat megoldása azon pontok halmaza, amelyekbe nem vezet be él, viszont kivezető élük van. A B részfeladat megoldása viszont fordítva: azon pontok halmaza, amelyeknek nincs kivezető élük, van viszont bemenő élük. A C részfeladat megoldása pedig azon pontok halmaza, amelyek maximális számú kivezető éllel rendelkeznek.

```
Rémhír(n,m,r,A,B,C):
  be:=(0,...,0); ki:=(0,...,0); max:=0, A:={}; B:={}; C:={}
  Ciklus i=1-től m-ig
    be(r(i,2)):=be(r(i,2))+1
    ki(r(i,1)):=ki(r(i,1))+1
    Ha ki(r(i,1))>max akkor max:=ki(r(i,1))
  Ciklus vége
  Ciklus i:=1-től n-ig
    Ha be(i)=0 és ki(i)>0 akkor A:=A∪{i}
  Ciklus vége
  Ciklus i:=1-től n-ig
    Ha be(i)>0 és ki(i)=0 akkor B:=B∪{i}
  Ciklus vége
  Ciklus i:=1-től n-ig
    Ha ki(i)=max akkor C:=C∪{i}
  Ciklus vége
Eljárás vége.
```

2. feladat: Hangya (13 pont)

A feladat megoldása azon múlik, hogy hogyan tudjuk jól kódolni a kocka csúcsaira azt, hogy honnan haladva melyik a balra, illetve jobbra levő pont. A köv tömb első 3 indexe x, y, illetve z lesz, a negyedik megadja, hogy a három szomszédjából melyikről van szó, az ötödik pedig, hogy az adott szomszéd melyik koordinátájáról. A szomszédokat úgy soroljuk fel, hogy ha egy pontba a j-edik szomszédjából jöttünk, akkor balra a j+1-edik, jobbra pedig a j+2-edik van, ciklikusan értve.

```
köv: tömb(0..1,0..1,0..1,0..2,0..2)=
        ((({0,0,0}((1,0,0),(0,1,0),(0,0,1)),
           {0,0,1}((0,0,0),(0,1,1),(1,0,1))),
          ({0,1,0}((0,0,0),(1,1,0),(0,1,1)),
           {0,1,1}((0,1,0),(1,1,1),(0,0,1)))),
         (({1,0,0}((0,0,0),(1,0,1),(1,1,0)),
           {1,0,1}((1,0,0),(0,0,1),(1,1,1))),
          ({1,1,0}((1,0,0),(1,1,1),(0,1,0)),
           {1,1,1}((1,0,1),(0,1,1),(1,1,0)))))

Hangya(x,y,z):
  x:=0; y:=0; z:=0; Ír(KiF,x,y,z)
  xx:=0; yy:=0; zz:=1; Ír(KiF,xx,yy,zz)
  Ciklus amíg nem vége(BeF)
    Olvas(BeF,c); j:=0
    Ciklus amíg nem (x=köv(xx,yy,zz)(j,0) és
                     y=köv(xx,yy,zz)(j,1) és
                     z=köv(xx,yy,zz)(j,2))
      j:=j+1
    Ciklus vége
```

```
    j:=j+1; Ha c='J' akkor j:=j+1
    j:=j mod 3; x:=xx; y:=yy; z:=zz
    xx:=köv(x,y,z)(j,0); yy:=köv(x,y,z)(j,1)
    zz:=köv(x,y,z)(j,2)
    Ír(KiF,xx,yy,zz)
  Ciklus vége
Eljárás vége.
```

3. feladat: Azonosító (15 pont)

Meg kell találni a kódban hátulról az első olyan betűt, amely mögött van az ábécében nála nagyobb. Ezekből a legkisebbet kell a helyére tenni, majd a maradékot ábécé sorrendben leírni!

```
Azon(régi,új):
  i:=hossz(régi); betű:=(hamis,...,hamis); utolsó:=régi(i)
  Ciklus amíg utolsó≤régi(i)
    betű(régi(i)):=igaz
    Ha régi(i)<utolsó akkor utolsó:=régi(i)
    i:=i-1
  Ciklus vége
  új:=régi(1..i-1); új:=új+utolsó; betű(régi(i)):=igaz
  Ciklus c='a'-tól 'z'-ig
    Ha betű(c) és c≠utolsó akkor új:=új+c
  Ciklus vége
Eljárás vége.
```

A feladat úgy is értelmezhető, hogy a bemenő szóban lehetnek azonos betűk. Ebben az esetben nem azt kell tárolni, hogy egy betű előfordult-e már, hanem azt, hogy hányszor fordult elő. Természetesen ennyiszer is kell betenni az új szóba..

```
Azon(régi,új):
  i:=hossz(régi); betű:=(0,...,0); utolsó:=régi(i)
  Ciklus amíg utolsó≤régi(i)
    betű(régi(i)):=betű(régi(i))+1
    Ha régi(i)<utolsó akkor utolsó:=régi(i)
    i:=i-1
  Ciklus vége
  új:=régi(1..i-1); új:=új+utolsó
  betű(utolsó):=betű(utolsó)-1; betű(régi(i)):=betű(régi(i))+1
  Ciklus c='a'-tól 'z'-ig
    Ciklus amíg betű(c)>0
      új:=új+c; betű(c):=betű(c)-1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

4. feladat: Rejtvény (15 pont)

Minden szóra minden lehetséges pozíción meg kell vizsgálni, hogy az adott szó ott kezdődhet-e a rejtvényben, illetve, hogy hányszor kezdődhet az adott pozíción!

```
Rejtvény(n,m,szó,mat,db):
  Ciklus k=1-től m-ig
    db:=0
    Ciklus i=1-től n-ig
      Ciklus j=1-től n-ig
        Számlál(szó(k),i,j,db,mat)
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Állítsuk össze minden irányban az adott pozícióban kezdődő szó hosszúságú karaktersorozatot, s ha ez egyenlő a kereset szóval, akkor számoljunk egyet!

```
Számlál(szó,i,j,db.mat):
  s:=('',...,'')
  Ciklus l=0-tól hossz(szó)-1-ig
    Ha j+l≤n akkor s(1):=s(1)+mat(i,j+l)
    Ha j-l>0 akkor s(2):=s(2)+mat(i,j-l)
    Ha i+l≤n akkor s(3):=s(3)+mat(i+l,j)
    Ha i-l>0 akkor s(4):=s(4)+mat(i-l,j)
    Ha i+l≤n és j+l≤n akkor s(5):=s(5)+mat(i+l,j+l)
    Ha i-l>0 és j+l≤n akkor s(6):=s(6)+mat(i-l,j+l)
    Ha i+l≤n és j-l>0 akkor s(7):=s(7)+mat(i+l,j-l)
    Ha i-l>0 és j-l>0 akkor s(8):=s(8)+mat(i-l,j-l)
  Ciklus vége
  Ciklus i=1-től 8-ig
    Ha s(i)=szó akkor db:=db+1
  Ciklus vége
Eljárás vége.
```

5. feladat: Ütemezés (17 pont)

Rendezzük a programokat a végrehajtási idő szerint nemcsökkenő sorrendbe! Ebben a sorrendben haladva vizsgáljuk a programokat. Az elsőt biztosan beválaszjuk. Legyen Mego=(p_1,…,p_m) az eddig beválasztott programoknak az ütemezés szerinti sorrendje, ami határidő szerint monoton nemcsökkenő! Az aktuális programot akkor válaszjuk be, ha beilleszthető az eddig beválasztottak küzé úgy, hogy minden program végrehajtása befejeződjün a határidejéig. Ez a következőképpen dönthető el. Legyen j a legnagyobb olyan index ($1 \le j \le m$), hogy a p_1,..p_j programok végrehajtási idejének összege nem nagyobb, mint az aktuális program határideje. Ha j+1-egikként beillesztve az aktuális programot minden program a határidejéig végrehajtódik a sorban, akkor beválaszjuk.

```
Ütemez(N,P):
  Rendez(P); m:=1; Mego(1):=1
  Ciklus i=2-től N-ig
    j:=1; ido:=0;
    Ciklus amíg j≤m és ido≤P(i).h;
      ido:=ido+P(j).v; j:=j+1
    Ciklus vége
    Ha ido+P(i).v≤P(i).h akkor
      Jo:=igaz; ido:=ido+P[i].v
      Ciklus jj=j-től m-ig
        ido:=ido+P(jj).v
        Ha ido>P(jj).v akkor Jo:=hamis; Kilép
      Ciklus vége
      Ha Jo akkor
        Ciklus jj=m-től -1 esével j-ig
          Mego(jj+1):=Mego(jj)
        Ciklus vége;
        Megi(j):=i; m:=m+1
      Elágazás vége
    Elágazás vége
  Ciklus vége
  Ki: m
  Ciklus i=1-től m-ig
    Ki: P(Mego(i)).azon, ' '
  Ciklus vége
Eljárás vége.
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Hírlánc (15 pont)

A hírláncot egy gráffal írhatjuk le. A gráf élein terjednek az üzenetek. A feladat szövegéből (*Minden ember akkor dönt arról, hogy melyik hírt fogadja el igaznak, amikor mindenkitől, aki neki hírt továbbíthat, megkapta a hírt.*) az következik, hogy ez a gráf körmentes.

Beolvasáskor minden pontnak számoljuk a befokát (`be`)! Számoljuk hogy mely híreket (`szavak(i).s`) és hányszor (`szavak(i).db`) hallottak azok, akiknek csak 0 befokú szomszédjuk van! A többségi hírt vagy a NINCS szót tároljuk minden ilyen elemre. Ezután ezek befokát 0-ra állítjuk. Ha ilyen elemet nem találunk, akkor befejezzük az eljárást.

```
Hírlánc(n,m,h,be,k):
  kell:=igaz
  Ciklus amíg kell
    i:=1
    Ciklus amíg i≤n és (be(i)=0 vagy nem Jó(i))
      i:=i+1
    Ciklus vége
    Ha i≤n akkor db:=0; maxdb:=1; több:=hamis
                 Ciklus j=1-től m-ig
                   Ha h(j,1)=i és szó(h(j,2))≠'NINCS'
                      akkor Gyűjt(szó(h(j,2)))
                 Ciklus vége
                 Ha nem több akkor szó(i):=szavak(maxdb).s
                 be(i):=0
    különben kell:=hamis
  Ciklus vége
Eljárás vége.
```

```
Gyűjt(s):
  j:=1
  Ciklus amíg j≤db és szavak(j).s≠s
    j:=j+1
  Ciklus vége
  Ha j≤db akkor szavak(j).d:=szavak(j).d+1
       különben db:=db+1; szavak(db).s:=s; szavak(db).d:=1
  Ha szavak(j).d>szavak(maxdb).d vagy db=1
     akkor maxdb:=j; több:=hamis
  különben ha szavak(j).d=szavak(maxdb).d akkor több:=igaz
Eljárás vége.
```

```
Jó(i):
  j:=1
  Ciklus amíg j≤m és (h(j,1)≠i vagy be(h(j,2))=0)
    j:=j+1
  Ciklus vége
  Jó:=(j>m)
Függvény vége.
```

2. feladat: Azonosító (15 pont)

Jelölje S(X) az $X=(x_1,..,x_n)$ karaktersorozat (permutáció) sorszámát, tehát azon permutációk számát, amelyek megelőzik X-et a lexikografikus rendezésben! Legyen P(X,i) azon Y permutációk száma, amelyek megelőzik X-et és Y-nak az első i-1 eleme megegyezik X első i-1 elemével! Tehát S(X) = P(X,1)+…+P(X,n-1). Legyen K(X,i) azon j>i indexek száma, amelyekre $x_j<x_i$!

Tehát P(X,i)=K(X,i)*(n-1)!, így

$$S(X) = \sum_{i=1}^{n-1} K(X,i) * (n-i)!$$

A feladat második részének, tehát a rákövetkező kód kiszámításához tegyük fel, hogy az $X=(x_1,..,x_n)$ a bemenetre a megoldás az $Y=(y_1,..,y_n)$ karaktersorozat, tehát Y az X rákövetkezője a lexikografikus rendezés szerint! A lexikografikus rendezés definíciója szerint van olyan i index ($1 \le i \le n$) index, hogy minden $j<i$-re $x_j=y_j$, és $x_i<y_i$. Mivel X-ben és Y-ban ugyanazok a karakterek ugyanannyi multiplicitással szerepelnek, ezért y_i megegyezik valamely x_j-vel, ahol $j>i$. Ha több ilyen is van, akkor a legikisebbnek kell lennie, mert Y a rákövetkező. Másrészről, ha több i indexre is teljesül, hogy van olyan $j>i$, hogy $x_i < y_i$., akkor a legnagyobb i-t kel választani. Ez azt jelenti, hogy i-re teljesül, hogy

$x_i<x_{i+1}>...>x_j>...>x_n$

$x_i<x_j$

$x_i>x_{j+1}$

Tehát az X rákövetkezőjét, Y-t úgy kapjuk, hogy x_i –t és x_j-t felcseréljük és i+1-től n-ig terjedő részsorozatot fordított sorrendben írjuk.

```
Azonosító(X):
  N:=Hossz(X); S:=0; P:=1
  Ciklus i=N-1-től
    P:=P*(N-i); K:=0
    Ciklus j=i+1-től N-ig
      Ha X(i)>X(j) akkor K:=K+1
    Ciklus vége
    S:=S+K*P
  Ciklus vége
  Ki: S; i:=N-1; Van:=hamis
  Ciklus amíg i>0 és X(i)>X(i+1)
    i:=i-1
  Ciklus vége
  Van:=i>0
  Ha i>0 akkor
    Y:=X; c:=X(i)
    Ciklus j=N-től -1 esével i+1-ig
      Ha X(j)>c akkor Y(i):=X(j); Y(i+N-j+1):=X(i); c:='z'
              különben Y(i+N-j+1):=X(j)
    Ciklus vége
  Elágazás vége
  Ha Van akkor Ki: Y különben Ki: 'NINCS'
Eljárás vége.
```

3. feladat: Ültetés (15 pont)

A bemenet egy $F:\{1,\ldots,n\}\to\{1,\ldots,n\}$ függvény, ami azt jelenti, hogy az x tanuló az F(x) ($x \neq F(x)$) tanuló mellé szeretne ülni.

Tekintsük az F függvény gráfját, amelynek pontjai az $1,\ldots,n$ számok, és irányított élei az (x,F(x)) párok! Jelölje Befok(x) az x pont befokát, tehát azon y-ok számát, amelyekre F(y)=x! Legyen x olyan pont, amely nincs körben, Befok(x)=k>1 és bármely 0 befokú pontbol x-be vezető úton minden pont befoka 1!

Ha $F(y_1)=x, \ldots, F(y_k)=x$, akkor az $x, y_1, \ldots, y_k$ pontok közül legfeljebb 2 igénye elégíthető ki. Vegyük azokat a pontokat, amelyek valamely 0-befokú pontbol x-be vezető úton vannak! Ezeket le tudjuk ültetni úgy, hogy $k-1$ kivételével mindegyik igényét kielégítjük. Ezeket a pontokat törölve a gráfból, a maradék optimális ültetéséhez hozzávesszük a kitörölt pontok előbbi leültetését, akkor

a kiindulási feladat egy optimális megoldását kapjuk. Eztán a gráf megmaradt részét, amely körhöz kapcsolódó láncokból áll, hasonló módszerrel lehet feldolgozni.

A beolvasáskor előállítjuk a függvény gráfjának transzponáltját is, mert erre szükség van. Először meghatározzuk, hogy mely pontok vannak körben. Majd minden pontra mélységi bejárással kiszámítjuk, hogy a transzponált gráfban hány olyan leszármazottja van, amelynek fokszáma nagyobb egynél. A redukálás során egy lépésben egy olyan fában lévő pontokat rakunk sorba (ültetünk le), amelynek csak a gyökerében van elágazás.

```
Ultet(F,G,Ul,Befok,Befok1):
  Befok1:=Befok
  Ciklus x=1-től N-ig {körben lévő pontok meghatározása}
    Ha Befok(x)=0 akkor
      y:=x
      Ciklus amíg Befok1(F(y))=1
        y:=F(y); Befok1(y):=Befok1(y)-1
      Ciklus vége;
      Befok1(y):=Befok1(y)-1
  Ciklus vége
  Szin():=(Fehér,…,Fehér)
  Ciklus x=1-től N-ig
    Ha Szin(x)=Fehér akkor Bejár(x)
  Ciklus vége
  eszám:=0
  Ciklus x=1-től N-ig
    Ha Befok(x)=0 és Befok(x)>1 és E(x)=0 akkor
      eszám:=eszám+1; Ecsomo(eszám):=x
  Ciklus vége
  szék:=1
  Ciklus amíg eszám>0
    x:=Ecsomó(1); Ecsomó(1):=Ecsomó(eszám); eszám:=eszám-1
    y:=G(x,1)
    Ciklus amíg Befok(y)>0
      y:=G(y,1)
    Ciklus vége
    Ciklus amíg
      Ul(y):=szék; szék:=szék+1; Befok(y):=0; y:=F(y)
    Ciklus vége
    Ul(x):=szék; szék:=szék+1
    Ciklus i=2-től Befok(x)-ig
      y:=G(x,i)
      Ciklus
        Ul(y):=szék; szék:=szék+1
        Ha Befok(y)=1 akkor y:=G(y,1) különben y:=0
      amíg y=0
    Ciklus vége
    Befok(x):=0; y:=F(x); i:=1;
    Ciklus amíg G(y,i)≠x
      i:=i+1
    Ciklus vége
    G(y,i):=G(y,Befok(y)); Befok(y):=Befok(y)-1
    Ciklus amíg Befok1(y) ≠1
      E(y):=E(y)-1
      Ha E(y)=0 és Befok(y)>1 akkor eszám:=eszám+1
                                    Ecsomó(eszám):=y
      y:=F(y)
    Ciklus vége
  Ciklus vége {eszám>0}
```

```
  Ciklus x=1-től N-ig
    Ha Befok(x)=1 akkor
      y:=x; Befok1(x):=0
      Ciklus
        Ha Befok(y)=1 akkor
          Ciklus
             Ul(y):=szék; szék:=szék+1;
             y:=F(y); Befok1(y):=0;
          amíg Befok1(y)=1 vagy x=y
          Ciklus vége
          Ha y=x akkor kilép
          Ul(y):=szék; inc(szék)
          Ciklus i=1-től Befok(y)-ig
            z:=G(y,i)
            Ciklus amíg z<>0 és Befok1(z)<>1
              Ul(y):=szék; szék:=szék+1
              Ha Befok(z)=0 akkor z:=0 különben z:=G(z,1)
            Ciklus vége
          Ciklus vége
        különben
          i:=1
          Ciklus amíg Befok1(G(y,i))=1
            i:=i+1
          Ciklus vége
          z:=G(y,i)
          Ciklus amíg Befok(z)>0
            z:=G(z,1)
          Ciklus vége
         Ciklus amíg z≠y
            Ul(z):=szék; szék:=szék+1; z:=F(z)
         Ciklus vége
         Ul(z):=szék; szék:=szék+1
         Ciklus i=1-től Befok(y)-ig
           u:=G(y,i)
           Ciklus amíg z≠u és u≠0 és Befok1(u)≠1
             Ul(u):=szék; szék:=szék+1
             Ha Befok(u)=0 akkor u:=0 különben u:=G(u,1)
           Ciklus vége
        Ciklus vége
      amíg y≠x
      Ciklus vége
    Elágazás vége
  Ciklus vége
Eljárás vége

Bejár(x):
  Szin(x):=Fekete; hany:=0
  Ciklus i=1-től Befok(x)-ig
    y:=G(x,i)
    Ha Szin(y)=Fehér akkor hany:=hany+Bejár(y)
                   különben hany:=hany+E(y)
  Ciklus vége
  E(y):=hany
  Ha Befok(x)>1 akkor hany:=hany+1
  Bejár:=hany
Eljárás vége
```

4. feladat: Szójáték (15 pont)

Legyen `E(j)` a `Z(1..j)` utolsó előfordulása `X`-ben, `U(j)` pedig a `Z(j)` betű utolsó előfordulása!

```
Szójáték(N,M):
  Mini:=0; Minh:=MaxN+1
  E:=(0,...,0); U:=(0,...,0); U(0):=1
  Ciklus i=1-től N-ig
    Olvas(BeF,X)
    Ciklus j=M-től 1-ig -1-esével
      Ha X=Z(j) és U(j-1)>0
        akkor U(j):=i
              Ha j=1 akkor E(j):=i különben E(j):=E(j-1)
    Ciklus vége
    Ha U(M)>0 és U(M)-E(M)<Minh
       akkor Minh:=U(M)-E(M); Mini:=E(M)
  Ciklus vége
Eljárás vége.
```

5. feladat: Körutazás (15 pont)

Szélességi bejárással haladjunk a kiinduló K pontból, tárolva, hogy melyik pontba honnan jöttünk! Ha a P→Q él vizsgálatakor Q-ban már jártunk, akkor ellenőrizzük, hogy csak a K pont közös őse P-nek és Q-nak! Ha igen, akkor egy kívánt kört találtunk. Ha ennek hossza kisebb, mint az eddig talált kör hossza, akkor megjegyezzük a P és Q pontokat, amelyekből a végén ki tudjuk iratni a megoldást adó körutat.

```
Számol(K,P1,P2):
  Üresít(S); Táv:=(Inf,...,Inf); Apa:=(0,...,0)
  Táv(K):=0; Sorba(S,K); M:=N+1
  Ciklus amíg NemÜres(S)
    Sorból(S,P); Újtáv:=Táv(P)+1, i:=1
    Ciklus amíg i≤Fok(P) és kell
      Q:=G(P,i)
      Ha Táv(Q)≥Inf akkor Apa(Q):=P; Táv(Q):=Újtáv; Sorba(S,Q)
      különben ha Q≠Apa(P) akkor R:=P
                                 Ciklus amíg R≠K
                                   Ut1(R):=igaz; R:=Apa(R)
                                 Ciklus vége
                                 R:=Q
                                 Ciklus amíg Ut1(R)=hamis
                                   R:=Apa(R)
                                 Ciklus vége
                                 R:=P
                                 Ciklus amíg R≠K
                                   Ut1(R):=hamis; R:=Apa(R)
                                 Ciklus vége
                                 Ha R=K és D(P)+D(Q)<M akkor
                                    P1:=P; P2:=Q; M:=D(P)+D(Q)
      Elágazások vége
      i:=i+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```