Programozási

II.

Nemes Tihamér
Nemzetközi Informatikai Tanulmányi Verseny
1995-99

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:

Ali György (ELTE), Frigó József (BME), Gulyás László (ELTE), Horváth László (ELTE), Klimkó Gábor (MTA ITA), Mohay Tamás (Oracle Hungary), Orbán Anna (Államigazgatási Főiskola), Papp Zoltán (KLTE), Reé Balázs (BME), Székely Jenő (ELTE), Szlávi Péter (ELTE), Tagányi György (BME), Török Turul (ELTE).

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 2* című kötet alapján készült.

ISBN 978-963-489-037-9

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

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

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, ill. a 16-19 éves tanulók (azaz a középiskolák I.-II., ill. 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., ill. 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. 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 az 1995-1999 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

1995. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Logo (36 pont)

Mit rajzolnak a következő – végtelen sokáig futó – LOGO programok?

A programban használt utasítások magyarázata a következő:

`FORWARD hossz`	előre lép az aktuális irányban hossz egységgel,
`LEFT szög`	balra fordul szög fokkal,
`RIGHT szög`	jobbra fordul szög fokkal,
`UP`	felemeli a tollát, azaz ettől kezdve mozgás közben nem rajzol,
`DOWN`	leteszi a tollát, azaz ettől kezdve mozgás közben rajzol,
`REPEAT db [...]`	a zárójelben levőket db-szer megismétli,
`TO`	eljárás eleje,
`END`	eljárás vége.

A.
```
TO mitrajzol A
   REPEAT 4 [FORWARD A LEFT 90]
   mitrajzol A/2
END
```

B.
```
TO mitrajzol A
   REPEAT 5 [FORWARD A LEFT 90]
   mitrajzol A/2
END
```

C.
```
TO mitrajzol A
   REPEAT 4 [FORWARD A LEFT 90]
   UP FORWARD A/4 LEFT 90 FORWARD A/4 RIGHT 90 DOWN
   mitrajzol A/2
END
```

2. feladat: Csere (20 pont)

A következő algoritmus véletlenszerűen cserélgeti az (1,2,3,4) elemeket tartalmazó A vektor elemeit.

```
Ciklus i=1-től 3-ig
  Ha Véletlenszám<1/2 akkor Csere(A(i),A(i+1))
Ciklus vége
```

A véletlenszám egy 0 és 1 közötti véletlenszerűen választott valós számot jelent. Add meg, hogy a következő sorrendű eredményekből melyik állítható elő:

A. 2,3,4,1 B. 3,2,1,4 C. 2,3,1,4 D. 2,4,1,3

E. Add meg, hogy milyen sorrendű eredmény nem állítható elő ezzel az algoritmussal!

3. feladat: Síknegyed (20 pont)

A következő algoritmus a sík egy tetszőleges (X,Y) pontjáról megadja, hogy a síkon hol található. Néhány esetben azonban nem működik helyesen, melyek ezek?

```
Síknegyed(X,Y):
  Ha X>0 és Y>0 akkor Ki: "I. síknegyed"
  Ha X<0 és Y>0 akkor Ki: "II. síknegyed"
  Ha X<0 és Y<0 akkor Ki: "III. síknegyed"
  Ha X>0 és Y>0 akkor Ki: "IV. síknegyed"
  Ha X=0 és Y=0 akkor Ki: "Origó"
Eljárás vége.
```

4. feladat: Útkereszteződés (24 pont)

Egy útkereszteződésben autók áthaladására a KRESZ háromféle szabályt alkalmaz:

- A főútvonalon haladónak mindig elsőbbsége van a mellékútvonalon haladóval szemben (két főútvonal sohasem találkozhat).
- Egyenrangú utak esetén az azonos úton (szemben) közlekedők esetén az egyenesen haladónak elsőbbsége van a kanyarodókkal szemben.
- Egyenrangú utak esetén a keresztező útvonalon haladók esetén annak van elsőbbsége, aki a másik jobbkeze felőli irányból érkezik.

Az autók mozgására három algoritmust írtunk. Add meg, hogy közülük melyik milyen esetben sérti meg a KRESZ előírásait, illetve van-e bennük felesleges vizsgálat!

```
A. Autó1:
     Előre a kereszteződésig
     Várj amíg van autó jobbról
     Előre
   Eljárás vége.
```

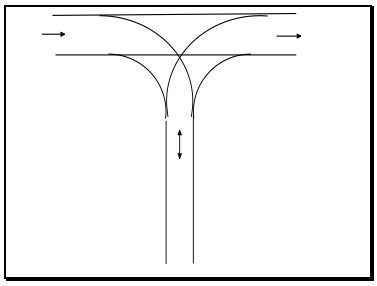
```
B. Autó2:
     Előre a kereszteződésig
     Várj amíg van autó szemből
     Balrafordulás
     Előre
   Eljárás vége.
```

```
C. Autó3:
     Előre a kereszteződésig
     Ha a keresztező főút akkor Várj amíg van autó jobbról
     Jobbrafordulás
     Előre
   Eljárás vége.
```

Kilencedik-tizedik osztályosok

1. feladat: Kitérő (12 pont)

Egy vasúti rendező-pályaudvar egyetlen kitérőt tartalmaz, az ábrának megfelelően.



Ha egy vonatszerelvény érkezik az 1,2,3...N sorszámú kocsikkal, akkor minden egyes lépésben három lehetőség közül választhatunk: (először megy be az 1. kocsi, utoljára az N.)

- ÁT: A szerelvény elején levő kocsit a kijárat felé továbbítjuk.
- BE: A szerelvény elején levő kocsit a kitérőbe visszük.
- KI: A kitérő végén álló utolsó kocsit a kijárat felé küldjük.

Állapítsd meg, hogy ha a beérkező szerelvény az 1,2,3,4 kocsikból áll, akkor lehet-e a kimenő szerelvény a következő sorrendű! Az előállíthatóknál add meg előállításuk lépéseinek sorrendjét, a többinél pedig magyarázd meg, hogy miért nem lehetséges az előállításuk!

A. 4,3,2,1 B. 4,1,2,3 C. 1,4,3,2 D. 1,4,2,3

2. feladat: Logo (16 pont)

A következő LOGO program szimmetrikus ábrákat rajzol:

```
TO mitcsinál A B
  FORWARD A
  LEFT B
  mitcsinál A B
END
```

A programban használt utasítások magyarázata a következő:

`FORWARD hossz`	előre lép az aktuális irányban hossz egységgel,
`LEFT szög`	balra fordul szög fokkal a teknőc síkjában,
`TO`	eljárás eleje,
`END`	eljárás vége.

A szimmetria lehet kettes (pl. a pillangó szárnyai), hármas (a háromlevelű lóhere), négyes (ilyen a négylevelű lóhere) stb. A következő eljáráshívások eredménye milyen szimmetriát mutat?

```
A. mitcsinál 100 144
B. mitcsinál 100 160
C. mitcsinál 100 150
D. mitcsinál 100 96
```

3. feladat: Útkereszteződés (10 pont)

Egy útkereszteződésben autók áthaladására a KRESZ háromféle szabályt alkalmaz:

- A főútvonalon haladónak mindig elsőbbsége van a mellékútvonalon haladóval szemben (két főútvonal sohasem találkozhat).
- Egyenrangú utak esetén az azonos úton (szemben) közlekedők esetén az egyenesen haladónak elsőbbsége van a kanyarodókkal szemben.
- Egyenrangú utak esetén a keresztező útvonalon haladók esetén annak van elsőbbsége, aki a másik jobbkeze felőli irányból érkezik.

Az autók mozgására három algoritmust írtunk. Add meg, hogy közülük melyik milyen esetben sérti meg a KRESZ előírásait, illetve van-e bennük felesleges vizsgálat!

```
A. Autó1:
     Előre a kereszteződésig
     Várj amíg van autó jobbról
     Előre
   Eljárás vége.
B. Autó2:
     Előre a kereszteződésig
     Várj amíg van autó szemből
     Balrafordulás
     Előre
   Eljárás vége.
C. Autó3:
     Előre a kereszteződésig
     Ha a keresztező főút akkor Várj amíg van autó jobbról
     Jobbrafordulás
     Előre
   Eljárás vége.
```

4. feladat: Transzformáció (12 pont)

Egy grafikus nyelven a képernyővel a következő transzformációkat végezhetjük:

- Négyzet alakú rész forgatása 90 fokkal balra (FORGAT((x,y),(p,q)))
- Téglalap alakú rész tükrözése a téglalap X-tengelyére (XTÜKÖR((x,y),(p,q)))
- Téglalap alakú rész tükrözése a téglalap Y-tengelyére (YTÜKÖR((x,y),(p,q)))

Milyen transzformációkkal állítható elő az ábrán látható állapotból a következő néhány állapot?
Csak 2-lépéses megoldásokat keress!

A	B	C
D	E	F
G	H	I

A.

B	E	C
G	H	F
A	D	I

B.

B	A	C
E	F	I
H	D	G

Példa:

A	B	C
D	E	F
G	H	I

`XTÜKÖR((1,2),(2,3))`

A	E	F
D	B	C
G	H	I

5. feladat: Szavak (20 pont)

A LOGO programnyelv következő függvényeit mondatokra is és szavakra is lehet alkalmazni:

`FIRST X`	megadja az X szó első betűjét (vagy az X mondat első szavát)
`BUTFIRST X`	megadja az X szót az első betűje nélkül (vagy az X mondatot az első szava nélkül)

`LAST X`	megadja az X szó utolsó betűjét (vagy az X mondat utolsó szavát)
`BUTLAST X`	megadja az X szót az utolsó betűje nélkül (vagy az X mondatot az utolsó szava nélkül)

Az önálló betűk elé ~ jelet, a szavak elé " jelet kell tenni, a mondatokat pedig [.] jelek közé tesszük, s a mondaton belül a szavak elé nem kell kitenni az idézőjelet.

Példa:

```
~B                          FIRST "SZÓ → ~S,
"SZÓ                        BUTFIRST "SZÓ → "ZÓ
[RÖVID MONDAT]              FIRST [RÖVID MONDAT] → "RÖVID
[EZ EGY HOSSZABB MONDAT]    BUTFIRST [MONDAT] → []
```

Mit adnak meg a következő LOGO kifejezések? Add meg a konkrét értéket, valamint fogalmazd meg általánosan is a kapott eredményt!

```
A. FIRST BUTFIRST [FEKETE MACSKA]

B. BUTFIRST FIRST [VÉGIGÉRŐ FA]

C. LAST FIRST [LOGO PROGRAMNYELVŰ UTASÍTÁSOK]

D. BUTLAST BUTFIRST [MI LESZ EZ]
```

6. feladat: Mit csinál? (14 pont)

Egy Pascal-szerű programnyelven írtunk programrészletet az alábbiakban. Ezen a nyelven az elágazások és a ciklusok végét is az End alapszó jelzi.

A. Mit csinál az alábbi programrészlet?

B. Mire szolgálnak az **l**, az **m**, illetve az **n** változók?

```
l:=False;
For i:=1 TO 100 DO
   If a[i]>0 then
      If l then
              If a[i]<m then m:=a[i]; n:=i;
              End;
           else
              l:=TRUE; n:=i; m:=a[n];
      End;
   End;
   End;
```

7. feladat: Levelezés (16 pont)

Józsi és Pisti órán leveleznek. A levélküldést szigorú szabályok szerint bonyolítják le:

1. Egyszerre (egy papírlapon) vagy egy betűt lehet küldeni, vagy egy speciális jelet.
2. Józsinak először egy speciális jelet kell küldenie, ami azt jelenti, hogy indul a levelezés.
3. Pistinek erre egy másik speciális jellel kell válaszolnia, ami azt jelenti, hogy jöhet a levél. Ezután küldheti Józsi egyesével levelének betűit.
4. Egyszerre maximum 5 levél lehet útközben (ennyi gyerek ül közöttük), egymást nem előzhetik meg, s nem is veszhetnek el.
5. Ha Pisti nem tudja elolvasni Józsi egy betűjét, akkor bármikor visszaüzenhet egy speciális jelet a betű sorszámával kiegészítve – ezzel kér ismétlést. Bármely küldött jel olvashatatlanná válhat, más hibajelenség azonban nem történhet vele.

6. Józsi erre az ismétlés tényére utaló speciális jelet köteles küldeni, majd a nem értett betűtől folytatja a levél küldését.
7. Józsi a levél végén egy újabb speciális jelet küld, ez jelenti a befejezést.

Az algoritmusokban az ÜZENET változó akkor kap értéket, ha egy papírlap megérkezett, s megvizsgálása után elveszti értékét (azaz minden lapot csak egyetlenegyszer lehet megnézni).

A két levelező algoritmusa, melyben a speciális jeleket kisbetűs szavakkal jelöljük:

```
Józsi:
  Küld(beszéljünk); I:=1
  Várj amíg ÜZENET≠jöhet
  Ciklus amíg I≤hossz(LEVÉL)
     Küld(LEVÉL(I)); I:=I+1
     Ha ÜZENET=nemértem(J) akkor I:=J; Küld(ismétlem)
  Ciklus vége
  Küld(vége)
Eljárás vége.
```

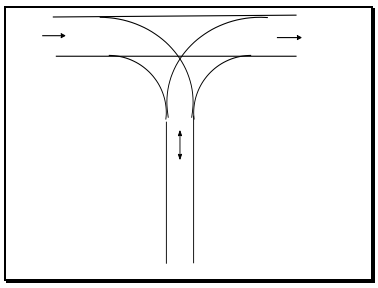
```
Pisti:
  Várj amíg ÜZENET≠beszéljünk
  Küld(jöhet); J:=1
  Várj amíg ÜZENET='' majd B:=ÜZENET
  Ciklus amíg B≠vége
    Ha B olvashatatlan akkor Küld(nemértem(J))
                             Várj amíg ÜZENET≠ismétlem
                      különben VETT(J):=B; J:=J+1
    Várj amíg ÜZENET='' majd B:=ÜZENET
  Ciklus vége
Eljárás vége.
```

Milyen konfliktushelyzetek adódhatnak elő a két algoritmus alapján? (Lehetséges, bár az algoritmusok alapján hibás válasz lenne a következő: Józsi nem várja meg Pisti *jöhet* üzenetét.)

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rendezőpályaudvar (12 pont)

Egy vasúti rendező-pályaudvar egyetlen kitérőt tartalmaz, az ábrának megfelelően.



Ha egy vonatszerelvény érkezik az 1,2,3...N sorszámú kocsikkal, akkor minden egyes lépésben három lehetőség közül választhatunk: (először megy be az 1. kocsi, utoljára az N.)

- ÁT: A szerelvény elején levő kocsit a kijárat felé továbbítjuk.
- BE: A szerelvény elején levő kocsit a kitérőbe visszük.
- KI: A kitérő végén álló utolsó kocsit a kijárat felé küldjük.

Állapítsd meg, hogy ha a beérkező szerelvény az 1,2,3,4 kocsikból áll, akkor lehet-e a kimenő szerelvény a következő sorrendű! Az előállíthatóknál add meg előállításuk lépéseinek sorrendjét, a többinél pedig magyarázd meg, hogy miért nem lehetséges az előállításuk!

A. 4,3,2,1 B. 4,1,2,3 C. 1,4,3,2 D. 1,4,2,3

2. feladat: Kártya (12 pont)

Egy kártyajátékos N lapot kap, de közülük egyszerre csak K lapot tud a kezében tartani. A következő algoritmusokban A(i) jelöli a kártyajátékos kezében levő i. lapot (1≤i≤K), mindhárom azt a tevékenységet játssza le, amelyben a kártyajátékosnak a j. sorszámú lapra van szüksége (1≤j≤N) –

ha a kezében van, akkor stratégiája szerint átrendezheti a kezében levő lapokat; ha nincs, akkor le kell tennie valamilyen stratégia alapján a kezében levő lapok közül egyet, majd a helyére felveheti a szükséges lapot. Ezt az eljárást természetesen a kártyajáték során akárhányszor végrehajthatja. Add meg, hogy a (*)-gal jelölt részre beírt egyes algoritmusokban milyen stratégiát alkalmaz a játékos!

```
  i:=1
  Ciklus amíg i≤K és A(i)≠j
     i:=i+1
  Ciklus vége
  (*)
```

```
A.   Ha i>K akkor Tedd le A(K)-t; i:=K
     Ciklus p=i-től 2-ig
        A(p):=A(p-1)
     Ciklus vége
     A(1):=j
```

```
B.   Ha i>K akkor Tedd le A(K)-t; A(K):=j; B(K):=1
     különben p:=i-1; B(i):=B(i)+1
               Ciklus amíg p>0 és B(p)>B(p+1)
                 Csere(A(p),A(p+1)); Csere(B(p),B(p+1))
                 p:=p-1
               Ciklus vége
     Elágazás vége
```

```
C.   Ha i>K akkor Tedd le A(K)-t
                   Ciklus p=K-tól 2-ig
                  A(p):=A(p-1)
                   Ciklus vége
                   A(1):=j
     Elágazás vége
```

3. feladat: Szöveg (10 pont)

Van egy *SZÖVEG* típusunk, megfelelő műveletekkel. Ezek a műveletek az alábbiak:

HOSSZ	a szöveg hosszát megadó függvény
KIIR	a képernyőre írja a megfelelő betűt, vagy szöveget
ELEJE	eljárás, amely paramétereként kapott szövegből levágja az utolsó karaktert
VEGE	eljárás, amely paramétereként kapott szövegből levágja az első karaktert
ELSO	a szöveg első karakterét adó függvény
UTOLSO	a szöveg utolsó karakterét adó függvény

A. Milyen sorrendben írja ki az alábbi két eljárás a paraméterként kapott szöveget, ha meghívjuk az Elj2 eljárást?

B. Mi lesz az eredmény az Elj2("ALMA") eljáráshívás után?

```
PROCEDURE Elj1(S:Szoveg);
BEGIN
  IF HOSSZ(S)>0 THEN Kiir(Utolso(S)); ELEJE(S); Elj2(S);
  END IF;
END Elj1;
```

```
PROCEDURE Elj2(S:Szoveg);
BEGIN
  IF HOSSZ(S)>0 THEN Kiir(Elso(S)); VEGE(S); Elj1(S);
  END IF;
END Elj2;
```

4. feladat: Telefonszámok (9 pont)

Egy rövidített hívószámok használatára alkalmas telefon leírása a rövidítés programozására így szól:

1. Nyomd le a STORE gombot!
2. Távolsági hívás esetén tárcsázd a 06-ot, majd nyomd le a PAUSE gombot, s utána a körzet 1-, vagy 2-jegyű körzetszámát!
3. Tárcsázd a 6- vagy 7-jegyű telefonszámot!
4. Nyomd le az AUTO gombot!
5. Tárcsázd azt az egyetlen számot, amit a későbbiekben rövidített számként szeretnél használni a teljes hívószám tárcsázása nélkül!

A telefonkészülék az alábbi algoritmus szerint működik (a STORE gomb megnyomása után kezdődik az algoritmus és a TÁBLA tömb azon elemébe teszi be a rövidített telefonszámot, amit utoljára nyomunk le – ez lesz a rövidítésre használt szám):

```
Telefonprogram:
  Be: T(1),T(2); Db:=2
  Ha T(1)=0 és T(2)=6 akkor Ciklus
                        Be: T(3)
                        amíg T(3)≠PAUSE
                        Ciklus vége
  Elágazás vége
  Be: X
  Ciklus amíg X≠AUTO
     Db:=Db+1; T(Db):=X; Be: X
  Ciklus vége
  Be: X; TÁBLA(X):=T
Eljárás vége.
```

Sorolj fel olyan eseteket, amikor vagy az algoritmus működik hibásan – és ilyenkor add meg a hibajelenséget is, vagy a tárolt telefonszám nem felel meg a Magyarországon használt – s a leírásban megadott – telefonszámoknak!

5. feladat: Transzformáció (8 pont)

Egy grafikus nyelven a képernyővel a következő transzformációkat végezhetjük:

- Négyzet alakú rész forgatása 90 fokkal balra (FORGAT((x,y),(p,q)))
- Téglalap alakú rész tükrözése a téglalap X-tengelyére (XTÜKÖR((x,y),(p,q)))
- Téglalap alakú rész tükrözése a téglalap Y-tengelyére (YTÜKÖR((x,y),(p,q)))

Milyen transzformációkkal állítható elő az ábrán látható állapotból a következő néhány állapot?

A	B	C
D	E	F
G	H	I

Csak 2-lépéses megoldásokat keress!

A.

B	E	C
G	H	F
A	D	I

B.

B	A	C
E	F	I
H	D	G

Példa:

A	B	C
D	E	F
G	H	I

XTÜKÖR((1,2),(2,3))

A	E	F
D	B	C
G	H	I

6. feladat: Keresés (23 pont)

A következő algoritmusrészlet szövegben keres szöveget, az egyszerű keresést néhány ötlettel gyorsítva.

```
siker:=hamis; i:=1
Ciklus amíg i≤hossz(A)-hossz(B)+1 és nem siker
  j:=hossz(B)
  Ciklus amíg 1≤j és B(j)=A(i+j-1)
     j:=j-1
  Ciklus vége
  siker:=(j=0)
  Ha nem siker akkor i:=i+C(j)
Ciklus vége
```

A. A keresés külső ciklusa mikor áll le?

B. A belső ciklus mikor fejeződik be?

C. Mi a szerepe a C vektornak?

D. Próbálj megfogalmazni minél pontosabb szabályt a C vektor meghatározására, azaz próbáld kitalálni, hogyan lehetne C elemeibe 1-nél nagyobb számokat tenni!

7. feladat: Titkosítás (16 pont)

A. Hogyan titkosítja a Szöveg szöveget a következő programrészlet? Mi a szerepe a Kulcs-nak?

(Az angol ABC betűi: ABCDEFGHIJKLMNOPQRSTUVWXYZ.)

```
Kódoltszöveg:=''
I:=1
Ciklus amíg I≤Hossz(Szöveg)
  Ha Szöveg(I)≥'A' és Szöveg(I)≤'Z' akkor
     Újbetű:=Karakter(((Karakterkód(Szöveg(I))+
     Karakterkód(Kulcs((I-1 mod Hossz(Kulcs))+1))
     -2*Karakterkód('A')+1) mod 26)+Karakterkód('A'))
  különben
     Újbetű:=Szöveg(I)
  Elágazás vége
  Kódoltszöveg:=Kódoltszöveg+Újbetű;   I:=I+1
Ciklus vége
```

Hossz:	a bemenetként kapott szöveg hosszát adja meg,
Karakter:	a megadott kódú karaktert adja,
Karakterkód:	a megadott karakter kódját adja meg,
Szöveg(I), Kulcs(I):	a Szöveg, Kulcs I. betűjét adja meg

A betűk kódjai követik az ABC-sorrendet (angol ABC).

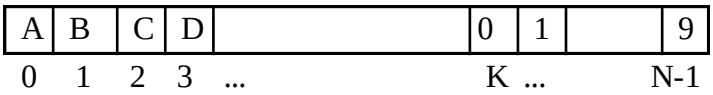
B. Mi volt az eredeti szöveg, ha a program eredményként ezt adta:

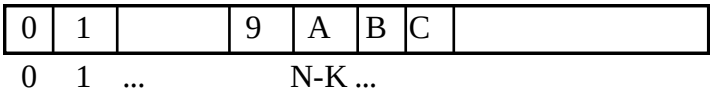
'SEG GSR SXVJM'

és a Kulcs értéke 'NEMES' volt?

8. feladat: Csere (10 pont)

Van egy tömbünk, amelyben az első K elemből álló részt, és a maradék N-K elemből álló részt fel kell cserélnünk úgy, hogy a két részen belül az elemek sorrendje ne változzon.





A feladatot megoldó eljárás:

```
procedure Felcserél(E,K,V: integer);
var
  I,J: integer;
begin
  I:=K-E;
  if I>V-K then I:=V-K;
  if I>0 then
  begin
    for J:=0 to I-1 do Csere(Tömb[E+J],Tömb[V-I+J]);
    if K-E>V-K then Felcserél(E,K,V-I)
               else Felcserél(E+I,K,V);
  end;
end;
```

Hívása:

```
  Felcserél(0,K,N)
```

A. Maximum hányszor kerül a Csere eljárás meghívásra?

B. Az eljárásba azonban hiba csúszott, így nem azt teszi, amit szeretnénk. Mit tesz a kitűzött feladat megoldása helyett?

C. Találd meg a hibát a Felcserél eljárásban!

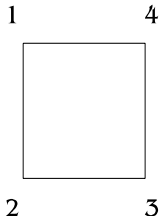
1995. Második forduló

Ötödik-nyolcadik osztályosok

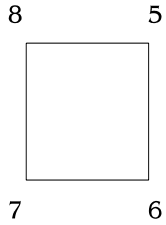
1. feladat: Kocka (16 pont)

Egy kocka csúcspontjait a következő szabályok szerint számozzuk meg:

- Az alsó lap csúcsaihoz tetszőleges pontból kiindulva az 1, 2, 3, 4 számokat rendeljük sorban az élek mentén



- A felső lap csúcsaihoz az 1-es fölötti csúcsból kiindulva, az előzővel azonos irányban haladva rendeljük a 8, 7, 6, 5 számokat sorban az élek mentén.



Készíts programot, amely beolvassa a csúcsok egy bejárási sorrendjét, majd eldönti, hogy csak a kocka élein haladva a csúcsok bejárhatók-e! Ha nem, akkor a program írja ki azokat a felsorolásban szomszédos csúcspárokat, amelyeket nem köt össze él! A beolvasott sorozatban biztosan 1 és 8 közötti egész számok vannak, s mindegyik pontosan egyszer.

2. feladat: Metró (32 pont)

Budapesten 3 (földalatti) metróvonal található, s mindegyiken sok-sok állomás. A három vonalnak egyetlen közös állomása van, ahol bármelyikről bármelyik másikra át lehet szállni (kihasználhatod, hogy a neve: Deák_tér). Egy külföldi turista áll az egyik metróállomáson, s egy másik metróállomásra akar eljutni. Készíts programot, amely beolvassa e két állomás nevét, majd megmondja, hogy a turistának az induló állomásról milyen irányba (melyik végállomás felé) hány megállót kell utaznia, s ha át kell szállnia, akkor ezt az átszállás előtti, illetve utáni metróvonalra is megadja. A létező metróállomások nevét megtalálhatod a METRO.PAS, METRO.BAS, METRO.LCN, METRO.LWR állományokban levő programrészletekben.

3. feladat: Állatok (27 pont)

Környezetünkben biológiai felmérést végeztünk, ún. táplálkozási párokat azonosítottunk (mi eszik mit?). E párok száma legfeljebb 30. A növények nem esznek semmilyen élőlényt, az állatok pedig vagy növényeket, vagy más állatokat esznek. A táplálkozási pár jelentése: az elsőnek megadott eszi a másodiknak megadottat, pl. "róka eszi fogoly", "csiga eszi fű". A táplálkozási párokban szereplő nevekben csak betűk lehetnek, s a neveket tetszőleges, általad meghatározott jel választhatja el egymástól

Készíts programot, amely a billentyűzetről beolvasott táplálkozási párok közül kiírja azoknak az állatoknak a nevét, amelyek esznek állatot (és esetleg növényt is)! Figyelem: ami nem eszik semmit, az növény.

Példa:

```
Bemenet:                Eredmény:

róka fogoly             róka
róka feketerigó         fogoly
fogoly földigiliszta    feketerigó
csiga fű
feketerigó csiga
földigiliszta avar
feketerigó gabonamag
```

Kilencedik-tizedik osztályosok

1. feladat: Fenyőfa (20 pont)

A Kísérleti Fanemesítő Intézet újfajta fenyőfákat nemesített ki. A fenyőfa törzséből pontosan 2 ág ágazik el, vagy egyetlenegy sem. Az egyes ágak ugyanolyan hosszúak és vastagok, mint a törzs, s a végükből legfeljebb újabb 2-2 ág ágazik el, vagy egy sem. Ezek megint ugyanolyan hosszúak, mint a törzs. Egy fát zárójelekkel és F betűkkel írunk le a számítógép számára: *(baloldali ág) F (jobboldali ág)* formában. A fának törzse biztosan van.

Példa:

ágnélküli fa:

F

kétágú fa:

(F) F (F)

bonyolultabb fa:

((F) F (F)) F (F)

Írj programot, amely meghatározza

A. a fa magasságát (a leghosszabb út hosszát a gyökértől valamelyik ág végéig) – a fenti három példában ez 1, 2, illetve 3;

B. a fa tömegét (feltételezve, hogy a törzs, illetve a vele azonos tömegű ágdarabok egységnyi tömegűek) – a fenti példában ez rendre 1, 3, illetve 5;

C. a fa elágazásainak számát – a fenti példában ez rendre 0, 1, illetve 2.

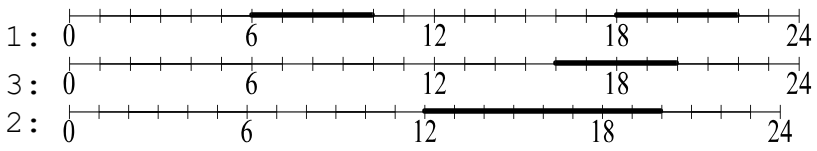
2. feladat: Televízió (20 pont)

Egy városban több televízióadó műsorát lehet fogni. Egy szöveges állományban tároljuk, hogy melyik mikor ad (feltesszük, hogy az adásidők a hét minden napján ugyanakkor vannak, a következő napra nem nyúlnak át, s egész órától egész óráig tartanak), egyes adók naponta többször is sugározhatnak műsort.

Az állomány minden sorában három szám található, egymástól egy-egy szóközzel elválasztva; az első az adó sorszáma, a második az adás kezdete, a harmadik pedig a vége (balról zárt, jobbról nyílt intervallumként). Az órák száma 0 és 24 közötti egész. Az állomány üres is lehet.

Példa:

```
1 18 22
1 6 10
3 16 20
2 12 20
```



Írj programot, amely

A. megadja a leghosszabb olyan időszakot egy napon belül, amikor az állományban tárolt adatok szerint egyetlen TV-adás sem fogható a városban (a fenti példában: 0-6);

B. meghatározza, hogy a nap melyik egyórás időszakában lehet a legtöbb műsor közül választani, s megadja ezek számát (a fenti példában: 18-19 vagy 19-20 a jó időszak, s ekkor 3 adást lehet nézni)!

3. feladat: Metró (20 pont)

Egy nagyvárosban 3 (földalatti) metróvonal található, s mindegyiken sok-sok állomás. A három vonalnak vagy egyetlen közös állomása, vagy pedig az 1.-2.-nak és a 2.-3.-nak külön átszállási helye van. Egy külföldi turista áll az egyik metróállomáson, s egy másik metróállomásra akar eljutni. Készíts programot, amely beolvassa e két állomás nevét, majd megmondja, hogy a turistának az induló állomásról milyen irányba (melyik végállomás felé) hány megállót kell utaznia, s ha át kell szállnia, akkor ezt az átszállás előtti, illetve utáni metróvonalra is megadja. A létező metróállomások nevét megtalálhatod a METRO.DAT állományban. (Az állományban soronként 1 adat szerepel, először az 1. vonal állomásainak száma, majd egyesével az állomások neve, utána a 2. vonal állomásainak száma ...) Az átszállóhely(ek) a közös név alapján ismerhető(k) fel.

4. feladat: Állatok (15 pont)

Környezetünkben biológiai felmérést végeztünk, ún. táplálkozási párokat azonosítottunk (mi eszik mit?). A növények nem esznek semmilyen élőlényt, az állatok pedig vagy növényeket, vagy más állatokat esznek. A BIO.INP állományban soronként egy-egy táplálkozási párt nevezünk meg, ahol a pár jelentése: az elsőnek megadott eszi a másodiknak megadottat, pl. „róka eszi fogoly”, „csiga eszi fű”. A két nevet egyetlen szóköz választja el. A BIO.INP állomány üres is lehet.

Készíts programot, amely kiválasztja a (csak) növényevő állatokat! Figyelem: ami nem eszik semmit, az növény.

Példa:

```
Bemenet:
róka fogoly
róka feketerigó
fogoly földigiliszta
csiga fű
feketerigó csiga
földigiliszta avar
feketerigó gabonamag
```

```
Eredmény:
csiga
földigiliszta
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Fenyőfa (15 pont)

A Kísérleti Fanemesítő Intézet újfajta fenyőfákat nemesített ki. A fenyőfa törzséből legalább 2 ág ágazik el, vagy egyetlenegy sem. Az egyes ágak ugyanolyan hosszúak, de feleakkora tömegűek, mint a törzs, s a végükből újra legalább 2-2 ág ágazik el, vagy egy sem. Ezek megint ugyanolyan hosszúak, mint amiből kinőttek, de feleakkora tömegűek. Egy fát zárójelekkel és F betűkkel írunk le a számítógép számára: *F (első ág) (második ág) ...(n. ág)* formában. A fának törzse biztosan van.

Példa:

ágnélküli fa:

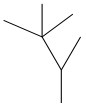
F

kétágú fa:

F (F) (F)

sokágú fa:

F (F) (F) (F) (F)



bonyolultabb fa:

F (F (F) (F)) (F) (F)

Írj programot, amely meghatározza

A. a fa magasságát (a leghosszabb út hosszát a gyökértől valamelyik ág végéig) – a fenti négy példában ez 1, 2, 2, illetve 3;

B. a fa tömegét (feltételezve, hogy a törzs egységnyi tömegű) – a fenti példában ez rendre 1, 2, 3, illetve 2.75;

C. a közös elágazásból induló ágak számának maximumát – a fenti példában ez rendre 0, 2, 4, illetve 3.

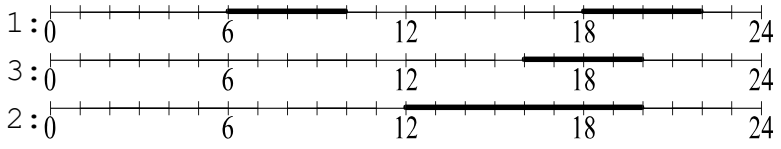
2. feladat: Televízió (16 pont)

Egy városban N (≥1) napon át több televízióadó műsorát lehet fogni. Egy szöveges állományban tároljuk, hogy melyiken mikor van adás. Egyes adók bármikor (akár többször is) sugározhatnak műsort, az adásidő egyik napról a másikra is átnyúlhat, sőt akár N napon át, megállás nélkül is tarthat.

Az állomány minden sorában hét szám található egymástól egy-egy szóközzel elválasztva, az első az adó sorszáma, a következő három az adás kezdete (napsorszám, óra, perc), az utolsó három pedig a vége (ugyanilyen jelentéssel) – az adásidő balról zárt, jobbról nyílt intervallumot jelent. Az órák száma 0 és 24, a percek száma 0 és 59 közötti egész. N értéke az állományban található napsorszámok alapján határozható meg. Az állomány üres is lehet.

Példa: (egy napon belüli, egész órakor kezdődő és végződő adásokkal)

```
1 1 18 0 1 22 0
1 1 6 0 1 10 0
3 1 16 0 1 20 0
2 1 12 0 1 20 0
```



Írj programot, amely

A. megadja a leghosszabb olyan időszakot, amikor az állományban tárolt adatok szerint egyetlen TV-adás sem fogható a városban (a fenti példában: 1. nap, 0.00– –6.00);

B. meghatározza, hogy az N-edik nap melyik percében lehet a legtöbb műsor közül választani, s akkor hány közül lehet (a fenti példában: 1. nap 18.00 és 19.59 között bármelyik perc jó, ekkor 3 adás fogható).

3. feladat: Sokszög (12 pont)

Készíts programot, amely a billentyűzetről tetszőleges sorrendben beolvassa egy konvex sokszög csúcsainak egész koordinátáit, majd kiírja őket olyan sorrendben, ahogyan a sokszög oldalai mentén bejárhatjuk őket az óramutató járásával ellentétes irányban! Kiindulópontnak a sokszög legkisebb x-koordinátájú csúcsát vedd (ha több ilyen van, akkor közülük a legkisebb y-koordinátájút). A koordináták biztosan helyesek, nem kell ellenőrizni őket.

A koordinátarendszer a szokásos, a csúcsok koordinátáját az (x,y) egész számpár adja meg, ahol x az abszcissza és y az ordináta. Az origó a (0,0) koordinátájú pont, x jobbra, y fölfelé nő.

Példa:

bemenő számsorozat: 2 -2 -2 4 -2 -3 1 2

értelmezése: (2,-2), (-2,4), (-2,-3), (1,2)

az eredmény: (-2,-3), (2,-2), (1,2), (-2,4)

4. feladat: Kutya (12 pont)

Egy kutya úgy úszik át a folyón a túlparton álló gazdájához, hogy minden pillanatban a gazdi irányába igyekszik. Ezt a mozgást kell közelítő módszerrel modellezned. A program számítsa ki, hogy a gazdájától milyen távolságra ér partot a kutya, és ez mennyi ideig tart! Ehhez a következő, valós értékű paramétereket kell beolvasnia a programnak a billentyűzetről:

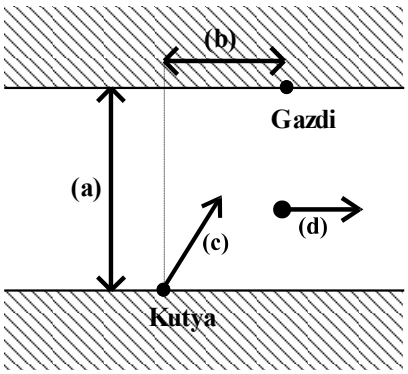
a) A folyó szélességét méterben.

b) A gazda távolságát méterben a kutya kezdőpontjának vetületétől a túlparton (pozitív, ha a folyásiránnyal azonos irányban van, negatív az ellenkező esetben).

c) A kutya sebességét (m/s, végig ugyanaz).

d) A folyó sebességét (m/s, mindenütt ugyanaz).

e) A közelítés pontosságát, azaz annak az időintervallumnak a hosszát másodpercekben, amelyen belül a program egyenes vonalú mozgással számolhat.



Grafikus ábrázolás nem szükséges, az értékelésnél nem veszszük figyelembe, a programod kipróbálását azonban segítheti.

A folyó két partját párhuzamos egyeneseknek tekintjük. A modellezés akkor álljon le, amikor a kutya már egy méternél közelebb kerül a túlsó parthoz.

A kutya mozgását haladási iránya, saját sebessége, valamint a folyó sebessége határozza meg. Mint tudjuk, mindkét sebesség állandó. A haladási irány, illetve a kutya sebességének a haladási iránytól függő x és y irányú összetevője azonban csak egy-egy időintervallumon belül tekinthető állandónak.

A kutya haladási irányát az alábbi képlettel számíthatjuk ki:

$$\text{Irány} = \text{ArcTan}\left(\frac{\text{GazdiYKoordináta - KutyaYKoordináta}}{\text{GazdiXKoordináta - KutyaXKoordináta}}\right).$$

(ArcTan: arkusz tangens függvény, megadja, hogy adott tangens érték mekkora szöghöz tartozik,

$$-\frac{\Pi}{2} \le ArcTan(x) \le \frac{\Pi}{2} \ .)$$

Egy időintervallum alatt a kutya x irányban:

(KutyaSebesség*Cos(Irány)+Vízsebesség)*Időintervallum hossza,

y irányban pedig:

KutyaSebesség*Sin(Irány)*Időintervallum hossza

utat tesz meg, hiszen a koordinátarendszert úgy célszerű megválasztani, hogy a víz az x-tengely mentén folyjon.

Példa:

paraméterek: (a): 200, (b): 100, (c): 5, (d): 6, (e): 1

eredmények: az eltelt idő: 157, a partot érés távolsága a gazdától kb. 206

(a távolság a közelítés miatt valós szám lesz, itt egy közelítő egész számot adunk meg)

5. feladat: Automata (20 pont)

Van egy gépünk, amely egy 40 jel hosszúságú *szalagból* és egy író-olvasó *fejből* áll. Jel egy maximum 40 elemű halmaz, az *ábécé* egy-egy eleme lehet. A gép maximum 30 különböző *állapotban* lehet.

A gép egy-egy jelet olvas a szalagról (onnan, ahol a fej van). A géphez tartozó szabálytáblázat írja elő, hogy a beolvasott jeltől és a gép pillanatnyi állapotától függően mit kell csinálni előbb a szalaggal, majd a fejjel, illetve mi lesz a gép következő állapota. Az elvégzendő művelet az ábécé egy elemének a szalagra írása, a fej jobbra, illetve balra mozgatása vagy helyben hagyása lehet.

Van a gépnek egy speciális állapota, a *végállapot*. Ha a gép ebbe az állapotba kerül, akkor az olvasott jeltől függetlenül leáll. Ha a szabálytáblázatban nincs a gép aktuális állapotára és az olvasott jelre

vonatkozó utasítás, akkor a gép automatikusan a végállapotba kerül. A gép indulásakor az író-olvasó fej a szalag 20. pozícióján áll (a sorszámozást az 1. pozíciótól kezdjük) és az 1-es sorszámú állapotban van.

Írj programot a fent leírt gép szimulálására!

A **GEP.INP** állomány első sora a szalagon levő jeleket tartalmazza (tehát pontosan 40 karakter hosszú). Az állomány további soraiban a szabálytáblázat elemei vannak, minden sorban egy szabály. A szabályok megadásának sorrendje tetszőleges. Egy *(állapot, jel)* párhoz csak egyetlen egy szabály tartozhat. (Ennek ellenőrzése **nem** szükséges.)

A szabályok formátuma: (a1,j1)::(a2,j2,*),

- ahol a1 az aktuális, a2 pedig az új állapot,
- j1 az aktuális állapotban olvasott, míg j2 a szalagon a helyébe írandó jel,
- a * pedig B, J, vagy – lehet. A B azt jelenti, hogy balra, a J azt, hogy jobbra kell mozgatni a fejet, míg – esetén nincs mozgatás.

Az állapotokat sorszámukkal jelöljük (a sorszámozás 1-től kezdődik), a végállapot sorszáma a **0**.

A. Ha a gép működése nem fejeződik be 1000 lépésen belül, akkor a **GEP.OUT** első sorába a **VEGTELEN** szót írjuk.

B. Ha a gép működése közben a fej elhagyja a szalagot, az első sorba értelemszerűen a **HIBA, BALRA KILEPETT** vagy a **HIBA, JOBBRA KILEPETT** üzenetet írjuk.

C. Ha a gép aktuális állapotára és az olvasott jelre a szabálytáblázatban nincs utasítás, akkor az első sorba a **NEMDEFINIALT** szót, a második sorba pedig az aktuális állapot sorszámát, egy szóközt és az olvasott jelet írjuk.

D. Ha a gép működése hibátlanul befejeződik, akkor a file első sorába a **VEGES** szót, a következő sorba pedig a szalag tartalmát írjuk.

1995. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mondd ki a számokat (23 pont)

Készíts programot, amellyel megtaníthatod a számítógépet a számok kimondására! A program olvasson be egy 1 és 100 közötti egész számot, majd – mivel a verseny helyszínén levő gépekben nincs hangkártya – a számot kiírja szövegesen a képernyőre!

Példa:

Ha ezt a számot adod be a programnak:	akkor ezt a szöveget írja ki eredménynek:
15	tizenöt
7	hét
99	kilencvenkilenc
100	száz

2. feladat: Autóbusz-menetrend (25 pont)

A Budapesti Közlekedési Vállalat (BKV) pontos menetrendet szeretne elhelyezni az 1-es autóbusz megállóiban, s ehhez kéri a Te segítségedet. Az autóbuszok 5 és 23 óra között közlekednek. Egy

buszvezető átlagos forgalom mellett feljegyezte az útvonal állomásain (maximum 20 állomás lehet) az egyes állomások közötti út megtételéhez szükséges időt, valamint (a végállomásokon kívül) az állomásokon a leszállással, felszállással eltöltött várakozási időt – mindegyiket egész számként. Készíts programot, amely a fentieken kívül beolvassa egy autóbusz indulási idejét a végállomásról, ezt ki is írja eredményként, majd megadja, hogy ez a busz mikor érkezik az egyes megállókba.

Példa:

Ha 5 állomás adatait olvassuk be, az autóbusz a közbülső állomásokon 1, 3, illetve 2 percet várakozik, az egyes állomások közötti menetideje rendre 5, 8, 3, valamint 8 perc, s az autóbusz a végállomásról 8 óra 10 perckor indul, akkor a kapott eredmény:

8 óra 10 indulás, 8 óra 15, 8 óra 24, 8 óra 30, 8 óra 40

Ha az autóbusz 8 óra 50 perckor indul, akkor a kapott eredmény:

8 óra 50 indulás, 8 óra 55, 9 óra 04, 9 óra 10, 9 óra 20

3. feladat: Lóverseny (15 pont)

Lóversenyeken a versenyrendezők igyekszenek az egyes lovaknak azonos esély adni a verseny megnyerésére. Emiatt a gyengébb lovak a rajtvonalról indulnak, az erősebbek pedig erősségük arányában egyre hátrábbról. Készíts programot, amely beolvassa N ló erősségét (1 és 100 közötti egész számok), s a következő szabályok alapján megadja, hogy a rajtvonaltól hány méter távolságra kell indulniuk:

- az 50, vagy annál kisebb erősségűek a rajtvonalról indulnak;
- a 90-nél nagyobb erősségűek a rajtvonal mögül 45 méterről indulnak;
- az 51 és 90 pont közöttiek erősségüktől függően a rajtvonaltól számított 5 és 40 méter között, 5 méterenként az erősségük arányában elosztva indulnak (tehát 51-től 55 pontig 5 méterre, 56-tól 60-ig 10 méterre, ...)

4. feladat: Furcsa egyszerűsítés (12 pont)

Írj olyan programot, amely felsorolja a következő tulajdonságokkal bíró `(X,Y)` számpárokat:

- `X` és `Y` (100 és 999 közötti) háromjegyű természetes számok a tízes számrendszerben ,
- `X` tízes számrendszerbeli alakja `ABC` (azaz `X = 100*A+10*B+C`),
- `Y` tízes számrendszerbeli alakja `CDE` (azaz `Y = 100*C+10*D+E`),
- az `X/Y` és `AB/DE` törtek megegyeznek (azaz az `X/Y = ABC/CDE` tört "egyszerűsíthető" a `C` számjegyek elhagyásával),
- `A`, `B`, `D` és `E` között nincs két egyforma számjegy.

Kilencedik-tizedik osztályosok

1. feladat: MIDI (25 pont)

A MIDI (Musical Instrument Digital Interface) számítógépek és szintetizátorok közötti adatcserére kidolgozott szabvány. A szabvány részben parancsokat tartalmaz a szintetizátor számára egy hang megszólaltatásának elkezdésére vagy befejezésére.

Ez a program például 3 hangot szólaltat meg egyszerre:

```
0 ON 60
0 ON 70
0 ON 80
10 OFF 60
10 OFF 80
10 OFF 70
10 ON 62
12 OFF 62
```

10 időegységig fog szólni a 60-as, a 70-es és a 80-as hang, majd 2 időegységig a 62-es. A program egy sora tehát a következő információkat tartalmazza: először azt az időpillanatot, amikor a parancsot végre kell hajtani, aztán magát a parancsot (ON esetén bekapcsolni, OFF esetén kikapcsolni kell a hangot), majd annak a hangnak a sorszámát, amire a parancs vonatkozik.

A MIDI programban háromféle probléma fordulhat elő:

a) Tekintsük az alábbi példát:

```
0 ON 60
10 ON 60
12 OFF 60
20 OFF 60
```

Ha meghallgatjuk ezt a programot, akkor nem fogunk két különálló hangot hallani, csak egyet, mégpedig 12 időegységig. Ezen úgy segíthetünk, hogy OFF parancsot illesztünk a programba 1 időegységgel a második ON parancs elé, az eredeti OFF parancsok közül pedig csak a másodikat hagyjuk meg. Így egy rövid szünetet hallunk a két hang között.

b) Egy másik probléma adódik akkor, ha ugyanazon időponthoz tartozó ON és OFF parancs is van a programban:

```
0 ON 60     0 ON 60
10 ON 60   10 OFF 60
10 OFF 60  10 ON 60
20 OFF 60  20 OFF 60
```

A bal oldali példában 10 időegységig lesz hallható a hang, a jobb oldaliban 20 időegységig, folyamatosan. (Ebben a példában a problémát okozó utasítások egymás mellett találhatók, de nem kizárt az sem, hogy legyen közöttük ugyanehhez az időponthoz tartozó más utasítás.)

A megoldás mindkét esetben ugyanaz: a problémát okozó OFF parancsot 1 időegységgel a megfelelő ON parancs elé kell áthelyezni. Ez is egy rövid szünetet eredményez a hang második megszólaltatása előtt.

c) Ha a program véget ér, és valamelyik hang még szól, akkor azt az utolsó időpont után 1 időegységgel ki kell kapcsolni.

Írj programot, ami tetszőleges számú MIDI programot beolvas a MIDI.BE nevű állományból, és a fenti változtatások elvégzése után kiírja a MIDI.KI állományba! Az egyes MIDI programokat olyan sorok választják el egymástól, amik csak a sor elején álló –1 számot tartalmazzák. Az utolsó program után –2 áll. A kimenet formátuma a bemenetével egyező.

Az időpont 0 és 65535 közötti egész szám lehet, a parancsokat (ON, OFF) mindig nagybetűvel írjuk, a hangok sorszámai 1 és 127 közötti egész számok. Egy soron belül az időpontot és a parancsot, valamint a parancsot és a hang sorszámát pontosan egy szóköz választja el.

A sorok az időpont szerinti nem csökkenő sorrendben vannak, de az egy adott időponthoz tartozó parancsok sorrendje tetszőleges.

A megoldás során feltehetjük, hogy kezdetben semmilyen hang nem szól, egy hiba kijavítása nem okoz újabb hibát, valamint hogy nincs hiba a nulladik időpontban.

Példa:

```
Bemenet         Kimenet
0 ON 60         0 ON 60
10 ON 60        9 OFF 60
12 OFF 60       10 ON 60
20 OFF 60       20 OFF 60
-1              -1
0 ON 60         0 ON 60
5 ON 70         5 ON 70
10 ON 60        9 OFF 60
10 OFF 60       10 ON 60
15 OFF 70       14 OFF 70
15 ON 70        15 ON 70
20 OFF 60       20 OFF 60
-2              21 OFF 70
                -2
```

2. feladat: Kirándulók (50 pont)

Pista barátunkat kinevezték egy turistaház igazgatójává. Feladata az, hogy az érkező csoportokat beossza a turistaház szobáiba. Az elosztás során természetesen figyelembe kell vennie a szobák befogadóképességét és a turisták igényeit. Segítsünk barátunknak, írjunk programot! Programunk legyen olyan, hogy szabadon meg lehessen adni, hogy a lehetséges feltételek, megszorítások közül melyeket követeljük meg.

Ennek megfelelően a program elején egy egyszerű menüből lehessen kiválasztani, hogy az alábbi feladatok közül melyiket akarjuk megoldani.

1. Megadott számú szobánk van, és mindegyiknek ismerjük a maximális befogadóképességét. Tudjuk ezen kívül azt, hogy hány kiránduló érkezik. (A kirándulókat és a szobákat egyaránt egytől kezdődő sorszámozással jelöljük.)
2. Mint **1.**, de ezúttal azt is tudjuk, hogy a kirándulók közül hányan lányok, illetve hányan fiúk (jelölés: a lányok sorszáma elé L-t, a fiúké elé F-et írunk). Fiúk és lányok nem kerülhetnek ugyanabba a szobába.
3. Mint 2., de a turistáknak lehetnek különleges kívánságaik is, azaz megmondhatják, hogy az adott párnak ugyanabban a szobában kell laknia.
4. Mint 3., de a turisták azt is megmondhatják, hogy csak olyannal kerülhessenek egy szobába, akiket felsoroltak, s a szobájukba mást ne tegyenek. (Ebben az esetben eltekinthetünk attól, hogy esetleg különböző neműek. Például házaspár, család, stb.)

Az 1. részfeladatban legfeljebb 1, a 2.-ban és a 3.-ban pedig legfeljebb 2 olyan szoba legyen, ami nincs teljesen feltöltve turistákkal, a 4. részfeladatban ezen kívül az egy szobába teendők mellett a lehető legkevesebb feltöltetlen hely maradjon!

A program bemenő adatait két szöveges állomány, a SZOBAx.TXT illetve a TURISTAx.TXT írja le. Az első értelemszerűen a rendelkezésre álló szobák adatait, míg a második a turistákról ismert információt tartalmazza. Az állomány nevében szereplő **x** egy 0 és 9 közötti tetszőleges számjegy lehet – az egy teszthez tartozó állományokat ez azonosítja.

A SZOBAx.TXT állomány formátuma:

Az állomány első sora megadja, hogy hány szoba van a turistaszállóban (maximum 10). Ezután minden sor egy-egy szoba befogadóképességét írja le. (A szobák sorszámozása az állománybeli sorrend szerint történik.)

Példa:

```
4
2
2
2
4
```

A TURISTAx.TXT állomány formátuma:

Az állomány első sora megadja, hogy hány turistáról van szó. A második sor a lányok számát tartalmazza. A lányok megegyezés szerint a kisebb sorszámokat kapják, azaz ha pl. 8 turista érkezik és ebből kettő lány, akkor őket L1-gyel és L2-vel jelöljük, míg a fiúkat a következők szerint: F3, F4, F5, F6, F7, F8.

Ezt egy olyan sor követi, mely azoknak a pároknak a számát tartalmazza, akiket egy szobában kell elhelyezni. Ezt a megfelelő számú, párokat tartalmazó sor követi. A példában a 3. szeretne egy szobában lakni az 5.-kel.

Végül az olyan csoportok száma következik, akik nem engednek maguk közé idegeneket, amit soronként követ az egyes csoportok leírása. A csoportok leírása egy darabszámmal kezdődik, majd szóközökkel elválasztva a csoportba tartozó személyek sorszámaival folytatódik. A példában az 1., a 7. és a 8. lehet például egy család, akik nem akarnak másokkal osztozni a szobájukon.

Példa:

```
8
2
1
3 5
1
3 1 7 8
```

A program eredménye:

A program eredménye az EREDMx.TXT állományba kerüljön, a következő formában. Az állomány minden sora egy szobát írjon le, az alábbiak szerint:

szobaszám: turista turista turista

Az állománynak a szobasorszámok szerint rendezettnek kell lennie. Amennyiben a feladat nem oldható meg, úgy az állomány egyetlen sort tartalmazzon, melyben a következő üzenet legyen olvasható: **NINCS MEGOLDAS**.

Példa:

```
1: L2
2: F3 F5
3: F4 F6
4: L1 F7 F8
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: MIDI (25 pont)

A MIDI (Musical Instrument Digital Interface) számítógépek és szintetizátorok közötti adatcserére kidolgozott szabvány. A szabvány részben parancsokat tartalmaz a szintetizátor számára egy hang megszólaltatásának elkezdésére vagy befejezésére.

Ez a program például 3 hangot szólaltat meg egyszerre:

```
0 ON 60
0 ON 70
0 ON 80
10 OFF 60
10 OFF 80
10 OFF 70
10 ON 62
12 OFF 62
```

10 időegységig fog szólni a 60-as, a 70-es és a 80-as hang, majd 2 időegységig a 62-es. A program egy sora tehát a következő információkat tartalmazza: először azt az időpillanatot, amikor a parancsot végre kell hajtani, aztán magát a parancsot (ON esetén bekapcsolni, OFF esetén kikapcsolni kell a hangot), majd annak a hangnak a sorszámát, amire a parancs vonatkozik.

A MIDI programban ötféle probléma fordulhat elő:

a) Tekintsük az alábbi példát:

```
0 ON 60
10 ON 60
12 OFF 60
20 OFF 60
```

Ha meghallgatjuk ezt a programot, akkor nem fogunk két különálló hangot hallani, csak egyet, mégpedig 12 időegységig. Ezen úgy segíthetünk, hogy OFF parancsot illesztünk a programba 1 időegységgel a második ON parancs elé, az eredeti OFF parancsok közül pedig csak a másodikat hagyjuk meg. Így egy rövid szünetet hallunk a két hang között.

b) Egy másik probléma adódik akkor, ha ugyanazon időponthoz tartozó ON és OFF parancs is van a programban:

```
0 ON 60     0 ON 60
10 ON 60   10 OFF 60
10 OFF 60  10 ON 60
20 OFF 60  20 OFF 60
```

A bal oldali példában 10 időegységig lesz hallható a hang, a jobb oldaliban 20 időegységig, folyamatosan. (Ebben a példában a problémát okozó utasítások egymás mellett találhatók, de nem kizárt az sem, hogy legyen közöttük ugyanehhez az időponthoz tartozó más utasítás.)

A megoldás mindkét esetben ugyanaz: a problémát okozó OFF parancsot 1 időegységgel a megfelelő ON parancs elé kell áthelyezni. Ez is egy rövid szünetet eredményez a hang második megszólaltatása előtt.

c) Ha a program véget ér, és valamelyik hang még szól, akkor azt az utolsó időpont után 1 időegységgel ki kell kapcsolni.

d) Ha OFF parancsot nem előz meg neki megfelelő ON parancs, akkor az OFF-ot el kell hagyni.

e) Ha ugyanahhoz az időponthoz több ON parancs is tartozik, akkor közülük csak egyet kell megtartani.

Írj programot, ami tetszőleges számú MIDI programot beolvas a MIDI.BE nevű állományból, és a fenti változtatások elvégzése után kiírja a MIDI.KI állományba. Az egyes MIDI programokat olyan sorok választják el egymástól, amik csak a sor elején álló –1 számot tartalmazzák. Az utolsó program után –2 áll. A kimenet formátuma a bemenetével egyező.

Az időpont 0 és 65535 közötti egész szám lehet, a parancsokat (ON, OFF) mindig nagybetűvel írjuk, a hangok sorszámai 1 és 127 közötti egész számok. Egy soron belül az időpontot és a parancsot, valamint a parancsot és a hang sorszámát pontosan egy szóköz választja el.

A sorok az időpont szerinti nem csökkenő sorrendben vannak, de az egy adott időponthoz tartozó parancsok sorrendje tetszőleges.

A megoldás során feltehetjük, hogy kezdetben semmilyen hang nem szól, egy hiba kijavítása nem okoz újabb hibát, valamint hogy nincs hiba a nulladik időpontban.

Példa:

```
Bemenet         Kimenet

0 ON 60         0 ON 60
10 ON 60        9 OFF 60
12 OFF 60       10 ON 60
20 OFF 60       20 OFF 60
-1              -1
0 ON 60         0 ON 60
5 ON 70         5 ON 70
10 ON 60        9 OFF 60
10 OFF 60       10 ON 60
15 OFF 70       14 OFF 70
15 ON 70        15 ON 70
15 OFF 80       20 OFF 60
15 ON 70        21 OFF 70
20 OFF 60       -2
-2
```

2. feladat: 80 nap alatt a Föld körül (75 pont)

Bizonyára ismered Verne Gyula regényét, melyben Phileas Fogg, az angol lord fogadásból 80 nap alatt körbeutazta a Földet. Ebben a feladatban az ő útját kell végigkövetned és segítened kell őt a fogadás megnyerésében.

Az egyes részfeladatok legyenek önállóan végrehajthatók egy menüből vezérelve.

1. rész: Szimuláció (25 pont)

A VAROSx.BE állomány az XIX. századi világtérkép néhány (≤10) nagyobb városát tartalmazza (nem feltétlenül azokat, amelyek az eredeti történetben szerepelnek, **x** egy 0 és 9 közötti számjegy, ez azonosítja az egy teszthez tartozó állományokat). Az állomány minden sora egy városra vonatkozó adatokat tartalmaz, egy-egy szóközzel elválasztva:

```
városnév hosszúsági_fok szélességi_fok
```

A hosszúsági, illetve szélességi fokokat az égtáj (K,É,NY,D), valamint a fok (0 és +180 közötti egész szám) azonosítja.

Példa:

```
London K0 É51
```

A JARATx.BE nevű állományban megtalálhatók a városok között igénybe vehető közlekedési járatok. Minden sora egy járat adatait tartalmazza egy-egy szóközzel elválasztva:

```
indulás cél eszköz első_járat menetidő várakozás díj
```

Példa:

```
New-York London hajó 3 5 1 5000
```

Az első járat indulása, a várakozás és a menetidő (egész) napban értendő az egyszerűség kedvéért az utazó Phileas Fogg saját idejében mérve (a 0 időpont a Londonból történő elindulás). A járatok oda-vissza közlekednek, a végpontokon valamennyit várakozva. A fenti példában ez azt jelenti, hogy New-Yorkból Londonba a 3., 15., 27,... napokon indul hajó, visszafelé pedig a 9., 21, 33,.. napokon. A járatok díja egységesen angol fontban van megadva. A járat mindig a két város közötti legrövidebb úton halad.

Phileas Fogg tervezett útját az UTITERVx.BE állomány tartalmazza:

```
cél eszköz
```

Példa:

```
Alexandria hajó
Szuez tevekaraván
```

Feladatod lépésenként végigkövetni Phileas Fogg útját, az eredményeket az UTx.KI állományba írva. Minden lépésben ki kell írnod az alábbi adatokat egy sorba:

```
cél eszköz érkezés készpénz
```

Példa:

```
London hajó 81 100
```

Az érkezési időpontot Phileas Fogg saját idejében mérve kell számítani. A készpénz Phileas Fogg maradék pénzét jelenti. Phileas Fogg, mint tudjuk 80000 font készpénzzel indul Londonból.

A szimuláció az alábbi üzenetekkel (állapotokban) érhet véget:

- **Nincs elég pénze** (az előírt utazás nem hajtható végre).
- **Visszaért Londonba: nem utazta körbe a földet** (ki kell találnod, hogy ezt hogyan állapítod meg).
- **Visszaért Londonba X>80 nap alatt: veszített.**
- **Visszaért Londonba X≤80 nap alatt: nyert.**

A körbeutazásnál ne feledkezz meg arról, hogy az abszolút (Londoni) idő Phileas Fogg saját idejéhez képest keleti irányú körbejárásnál 1 nappal kevesebbet, nyugati felé viszont 1 nappal többet jelez,

2. rész: Optimális útiterv (25 pont)

Készítsd el a VAROSx.BE és JARATx.BE adatainak ismeretében az optimális útitervet és írd ki az OPTTERVx.KI állományba. Ennek formátuma egyezzen meg az UTITERVx.BE formátumával.

Ha Phileas Fogg nem tud időben (80 nap alatt) célba érni, illetve útközben elfogy a pénze, akkor az állomány első sorába ki kell írni, hogy 'VESZTETT'.

Ha van 80 nap alatt célba vivő út, akkor közülük ki kell választani az optimálisat. Az optimális útiterv az alábbi követelményeknek kell, hogy eleget tegyen:

- Ha több útiterv is kínálkozik, melyben 80 napnál hamarabb célba érhet, akkor ezek közül azt kell választani, amelyik a legolcsóbb (Phileas Fogg-nak a legtöbb készpénze marad).
- Ha a készpénz megegyezik, akkor a leggyorsabb útitervet kell választani.

A verseny végeredménye:

I. kategória

1. Felföldi Zsolt	Fazekas Mihály Gimnázium, Budapest
2. Förhécz András	Teleki Blanka Gimnázium, Székesfehérvár
3. Szász Barnabás	Petőfi Sándor Általános Iskola, Debrecen
4. Szeti Balázs	Hétvezér téri Általános Iskola, Székesfehérvár
Kővári Kálmán	III. Béla Gimnázium, Baja
6. Jánossy Gergely	Váci u. Általános Iskola, Budapest
7. Török Gábor	Sellyei Körzeti Általános Iskola, Sellye
Petres Zoltán	Trefort Ágoston Gimnázium, Budapest
9. Szabó András	Ságvári Endre Gimnázium, Zalaegerszeg

10. Bódi Ádám	Földes Ferenc Gimnázium, Miskolc
II. kategória	
1. Elek Róbert	Kalmár László Szakközépiskola, Budapest
2. Hartmann Miklós	Petőfi Sándor Evangélikus Gimnázium, Bonyhád
3. Marhefka István	Avasi Gimnázium, Miskolc
4. Andics Árpád	Tóth Árpád Gimnázium, Debrecen
5. Nagy Attila	Vetési Albert Gimnázium, Veszprém
6. Ujhelyi Gábor	Földes Ferenc Gimnázium, Miskolc
7. Peller Balázs	Kalmár László Szakközépiskola, Budapest
8. Lövey László	Neumann János Szakközépiskola, Eger
9. Fazekas Dániel	Révai Miklós Gimnázium, Győr
Nagy András	Leőwey Klára Gimnázium, Pécs
Smulovics Péter	Berzsenyi Dániel Gimnázium, Budapest
III. kategória	
1. Kovács Gábor Zsolt	Lovassy László Gimnázium, Veszprém
2. Kovács Gábor	Radnóti Miklós Gimnázium, Budapest
Fige Péter	Herman Ottó Gimnázium, Miskolc
4. Tringel Mihály	Földes Ferenc Gimnázium, Miskolc
Bámer Balázs	Szent István Gimnázium, Budapest
6. Gosztolya Gábor	Ságvári Endre Gimnázium, Szeged
7. Blahut György Gábor	Szent István Gimnázium, Budapest
8. Lakatos Roland	Zrínyi Miklós Gimnázium, Zalaegerszeg
9. Késmárki László	Teleki Blanka Gimnázium, Székesfehérvár
10. Dávid Norbert	Révai Miklós Gimnázium, Győr
Szabó Balázs	Vetési Albert Gimnázium, Veszprém

1996. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rajzolgatás (25 pont)

Mit rajzolnak a következő LOGO-nyelvű programok? Méreteket is adj meg! (A teknőc kezdetben a képernyő közepén áll és az X-tengely irányába néz. Kezdetben a toll a papíron van, `PENUP` felemeli a papírról, `PENDOWN` leengedi a papírra. A teknőc `FORWARD x` hatására előre lép `x` egységgel, `REPEAT n [utasítások]` hatására pedig a zárójelbe tett utasításokat `n`-szer megismétli. Az `n` értéke a C részfeladatban 2-vel, a D részfeladatban pedig 3-mal osztható.)

```
A. REPEAT n [FORWARD 5 PENUP FORWARD 5 PENDOWN]
B. REPEAT n [FORWARD 1 PENUP FORWARD 5 PENDOWN]
C. REPEAT n/2 [FORWARD 1 PENUP FORWARD 5 PENDOWN
       FORWARD 5 PENUP FORWARD 5 PENDOWN]
D. REPEAT n/3 [FORWARD 1 PENUP FORWARD 5 PENDOWN FORWARD 5
       PENUP FORWARD 5 PENDOWN FORWARD 5 PENUP FORWARD 5
       PENDOWN]
```

2. feladat: Mit csinál? (24 pont)

Mit csinálnak a következő algoritmusok (`X` mindegyikben 0 és 255 közötti egész)?

A.
```
Be: X
Ciklus 8-szor
  A:=X mod 2
  Ki: A
  X:=X div 2
Ciklus vége
```

B.
```
Be: X; A:=0
Ciklus 8-szor
  A:=(A+(X mod 2)) mod
2
  X:=X div 2
Ciklus vége
Ki: A
```

C.
```
Be: X; A:=0
Ciklus 8-szor
  A:=A+(X mod 2)
  X:=X div 2
Ciklus vége
Ki: A
```

3. feladat: Szövegelés (18 pont)

Az alábbi 3 algoritmus a szöveg típusú `S` változót írja ki valamilyen átalakítás után egy 80 karakter szélességű sorba. Az `S` karaktereinek száma a futás elején legfeljebb 80 lehet. Melyik milyen formában írja ki a szöveget?

A.
```
Ciklus amíg hossz(S)<80
  S:=" "+S
Ciklus vége
Ki: S
```

B.
```
Ciklus amíg hossz(S)<80
  S:=S+" "
Ciklus vége
Ki: S
```

C.
```
Ciklus amíg hossz(S)<80
  Ha hossz(S) páros akkor S:=S+" "
                  különben S:=" "+S
Ciklus vége
Ki: S
```

4. feladat: Vacakánykészítő Művek (33 pont)

A Vacakánykészítő Művekben (a továbbiakban VM) 10 és 13 óra között kell a vacakányt előállítani. A vacakány részei, amint közismert: a lapostya, a kevenye és természetesen a szegentyű.

A lapostyát laposítani kell. Festeni csak a laposított lapostyát lehet. A festett lapostyát utólag sorszámmal látják el.

A kevenyét először jó erősen kevelni kell. A kevenyén likaknak kell lenni. Lakkozni a kevelt és likasztott kevenyét lehet. Lakkozás után minimum 30 perc pihentetés kell.

A szegentyűt hegyezik, majd festik.

A vacakány összeszerelésének menete: a késztermékgyártó üzemben a sorszámozott lapostyát és a pihentetett kevenyét a festett szegentyű segítségével kapcsolják össze.

Szerelés után iktatják és csomagolják a készárut (ezeket egyszerre is végezhetik).

Minden művelet fél óráig tart. 12 és 13 óra között minden munkásnak jár fél óra ebédidő.

A VM-ben hárman dolgoznak. Péter a szerelésen és az iktatáson kívül mindennel foglalkozhat. Jakab tud laposítani, kevelni; a szegentyűgyártó üzembe nem léphet be, de a késztermékkel minden műveletet elvégezhet. Károly vizsgázott sorszámozó, likasztó és hegyező, sőt csomagolhat és iktathat is.

A. Ha a munkát 10 órakor kezdik, mikor vihetik el legkorábban a gyárból a vacakányt?

B. Mikor ebédel Jakab?

C. Ki csomagolja a vacakányt?

D. A lapostyát vagy a szegentyűt festik-e előbb?

E. Hány munkafolyamatot végez Péter, Jakab, illetve Károly?

Kilencedik-tizedik osztályosok

1. feladat: Rajzolgatás (18 pont)

Mit rajzolnak a következő LOGO-nyelvű programok? Méreteket is adj meg! (A teknőc kezdetben a képernyő közepén áll és az X-tengely irányába néz. Kezdetben a toll a papíron van, `PENUP` felemeli a papírról, `PENDOWN` leengedi a papírra. A teknőc `LEFT s` hatására balra, `RIGHT s` hatására jobbra fordul `s` fokkal, `FORWARD x` hatására előre, `BACK x` hatására hátra lép `x` egységgel, `REPEAT n [utasítások]` hatására pedig a zárójelbe tett utasításokat `n`-szer megismétli. Az `n` értéke a B részfeladatban 2-vel osztható.)

```
A. REPEAT n [REPEAT 5 [FORWARD 5 LEFT 90] RIGHT 90 PENUP
             FORWARD 5 PENDOWN]

B. REPEAT n/2 [REPEAT 5 [FORWARD 5 LEFT 120]
               LEFT 120 PENUP FORWARD 5 PENDOWN
               REPEAT 5 [FORWARD 5 RIGHT 120]
               RIGHT 120 PENUP FORWARD 5 PENDOWN]

C. REPEAT n [REPEAT 8 [FORWARD 2 BACK 4 FORWARD 2 LEFT 45]
             PENUP FORWARD 8 PENDOWN]
```

2. feladat: Logikai műveletek (12 pont)

Az `N` (>1) elemű `A` vektorban egész számok vannak. Milyen vektorbeli értékek esetén írnak ki az alábbi programok `IGEN` választ?

A.

```
V:=Igaz
Ciklus i=2-től N-ig
   V:=V And (A(i)>A(i-1))
Ciklus vége
Ha V akkor Ki: "IGEN"
```

B.

```
V:=Hamis
Ciklus i=2-től N-ig
   V:=V Or (A(i)>A(i-1))
Ciklus vége
Ha V akkor Ki: "IGEN"
```

C. Van olyan eset, amikor a két program a vektorbeli elemek értékétől függetlenül egyformán viselkedik (mindkettő az `IGEN` eredményt írja ki, vagy egyik sem ír ki semmit). Melyik ez az eset?

3. feladat: Kördiagram (14 pont)

N db adatból – ezeket az A(1..N) tömbben tároljuk – olyan kördiagramot szeretnénk készíteni, amelyben az egyes körcikkek különböző színűek. A rajzoláshoz felhasználjuk a

```
Körcikk(szög1,szög2,i)
```

eljárást, amely egy origó középpontú, 100 egység sugarú körcikket rajzol. A körcikk sugarai az X-tengellyel szög1, illetve szög2 fokos szöget zárnak be, belsejének i kódú a színe.

A feladat megoldására készített eljárásban (Kördiagram) van néhány hiba.

```
Kördiagram:
  S:=0
  Ciklus I=1-től N-ig
     S:=S+A(I)
  Ciklus vége
  Ciklus I=1-től N-ig
     SZ:=A(I)/S*180; Körcikk(0,SZ,N)
  Ciklus vége
Eljárás vége.
```

A. Mit csinál hibásan az eljárás?

B. Javítsd ki a hibákat!

4. feladat: Levelek (19 pont)

Egy vállalat dolgozóinak a beosztása a következő lehet: igazgató, igazgatóhelyettes, osztályvezető, alkalmazott. Az alkalmazottak valamelyik osztályon dolgoznak. Az igazgató közvetlen főnöke az igazgatóhelyetteseknek, az igazgatóhelyettesek az osztályvezetőknek, az osztályvezetők a saját osztályuk alkalmazottjainak. Mindenki küldhet levelet a közvetlen főnökének és összes (nem csak közvetlen) beosztottjának. A dolgozók beosztását PROLOG-szerű *tényállításokkal* írjuk le:

Példa:

```
igazgató("Avar András").

igazgatóhelyettes("Álmos Ádám").
igazgatóhelyettes("Boldog Blanka").

osztályvezető("Császár Csaba", "Pénzügyi osztály").
osztályvezető("Demeter Dóra", "Tervosztály").

alkalmazott("Elekes Eszter", "Tervosztály").
alkalmazott("Forró Ferenc", "Tervosztály").
```

A levélküldés szabályait a következő – félkész – PROLOG-szerű program definiálja:

```
küldhet(Feladó,Címzett) {Feladó küldhet levelet Címzett-nek}
            ha közvetlenFőnöke(Feladó,Címzett) vagy
               beosztottja(Feladó,Címzett).

közvetlenFőnöke(A,F) ha .. {A-nak közvetlen főnöke F}.

beosztottja(F,A) ha ..  {F-nek közvetlen vagy közvetett beosztottja A}.
```

Fejezd be a két megkezdett eljárást! A ha alapszó utáni formulákban a nem, a vagy, valamint az és logikai műveleteket, továbbá a tényállításokat is használhatod a már definiált vagy általad definiálandó eljárások mellett.

5. feladat: Vasútmodell (18 pont)

Egy vasútmodell-szerelvényt az alábbi paranccsal mozgathatunk a sínen:

`Mozgat(Y,V)` – `Y` távolságot tesz meg, végsebessége `V` lesz.

A szerelvény csak egyenletesen tud gyorsulni vagy lassulni. Egy eljárást készítettünk, amely a szerelvényt `S` távolságra juttatja el (a ^ jel hatványozást jelöl):

```
Vezérlés(X,V):
  Y:=A*DT^2/2 + V*DT
  Ha X+Y≤S/2 és V+A*DT ≤W
     akkor Vúj:=V+A*DT
           Mozgat(Y,Vúj); Vezérlés(X+Y,Vúj); Mozgat(Y,V)
  különben Mozgat(S-2*X,V)
Eljárás vége.
```

Kezdetben `X=0`, `V=0`, `DT=1`, `A=2`, `S=20` és `W=10`! Másold le az alábbi táblázatot, és írd bele a `Mozgat` eljárás i-edik meghívása előtt (i = 1,2,3,...) érvényes értékeket!

i	1	2	3	4	5	6	7	8	9	10	11
X											
V											
Y											
Vúj											
megtett út	**0**										

A „megtett út” sorba az i-edik lépésig megtett útszakaszok összege írandó.

6. feladat: Csak párhuzamosan! (19 pont)

Egy számítógépben "nagyon sok" processzor van, így a kifejezések kiszámítását egyszerre több processzor végezheti.

Példa:

Az (a+b)*(c/d)/(e+f) kifejezés optimálisan három lépésben számolható ki két processzorral:

1. lépés: (a+b) és (c/d) egyidejű kiszámítása két processzorral – legyen a két részeredmény u és v,

2. lépés: u*v és (e+f) egyidejű kiszámítása két processzorral – legyen a két részeredmény x és y,

3. lépés: x/y kiszámítása egy processzorral.

Egy kifejezés párhuzamos kiszámítását akkor tekintjük optimálisnak, ha a lehető legkevesebb lépésben végezhető el úgy, hogy a párhuzamosan felhasznált processzorok száma minimális.

Állapítsd meg, hogy optimális esetben a következő kifejezések hány lépésben számíthatók ki, s kiszámításukhoz hány processzorra van szükség! (A zárójelezés nem bontható fel a kiszámítás gyorsítása érdekében!)

A. a+b*c+d*e

B. a*b+c*(d*e+f/g+h*i)

C. a*(b+c)-d*(e/f-g*h)+i*j

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Szövegelés (7 pont)

Az alábbi algoritmusrészlet a szöveg típusú S változó értékét alakítja át; S szavakat és a szavakat elválasztó egy-egy szóközt tartalmaz.

```
S:=S+" "
Ciklus amíg első(S)≠" "
  S:=elsőutániak(S)+első(S)
Ciklus vége
S:=elsőutániak(S)
```

A. Mi lesz S értéke az algoritmusrészlet végrehajtása után? (Gondolj a kivételes esetekre is!)

B. Mire szolgál a ciklus előtti, illetve utáni értékadás?

2. feladat: Mit csinál? (10 pont)

Az alábbi program A és B értékét határozza meg az N elemű X vektor értékei alapján. M értéke kezdetben 1.

```
  Mitcsinál(X,M,N,A,B):
    Ha N-M≤1 akkor Ha X(M)≤X(N) akkor A:=N; B:=M
                                különben A:=M; B:=N
    különben K:=(M+N) div 2
             Mitcsinál(X,M,K,A1,B1)
             Mitcsinál(X,K+1,N,A2,B2)
             Ha X(A1)>X(A2) akkor A:=A1 különben A:=A2
             Ha X(B1)<X(B2) akkor B:=B1 különben B:=B2
    Elágazások vége
  Eljárás vége.
```

A. Mi lesz A és B értéke?

B. Hányszor kell összehasonlítani az X tömb elemeit a megoldás során, ha N 2 valamilyen hatványa?

C. Milyen esetben lesz azonos a futás végén A és B értéke?

3. feladat: Mit számol? (10 pont)

Az X és Y tömbök egy sokszög egymást követő csúcsainak koordinátáit tartalmazzák. Az alábbi algoritmusrészlet A és B értékét határozza meg.

```
A:=0; B:=0; i:=N
Ciklus j=1-től N-ig
  L:=X(i)-X(j); K:=Y(i)-Y(j)
  A:=A+L*(Y(i)+Y(j)]); B:=B+négyzetgyök(L*L+K*K)
  i:=j
Ciklus vége
A:=abszolútérték(A/2)
```

A. Mi lesz az A és a B értéke?

B. Mire szolgál az abszolútérték-képzés?

4. feladat: Csak párhuzamosan! (16 pont)

Egy számítógépben „nagyon sok" processzor van, így a kifejezések kiszámítását egyszerre több processzor végezheti.

Példa:

Az (a+b)*(c/d)/(e+f) kifejezés optimálisan három lépésben számolható ki két processzorral:

1. lépés: (a+b) és (c/d) egyidejű kiszámítása két processzorral – legyen a két részeredmény u és v,

2. lépés: u*v és (e+f) egyidejű kiszámítása két processzorral – legyen a két részeredmény x és y,

3. lépés: x/y kiszámítása egy processzorral.

Egy kifejezés párhuzamos kiszámítását akkor tekintjük optimálisnak, ha a lehető legkevesebb lépésben végezhető el úgy, hogy a párhuzamosan felhasznált processzorok száma minimális.

Állapítsd meg, hogy optimális esetben a következő kifejezések hány lépésben számíthatók ki, s kiszámításukhoz hány processzorra van szükség! (A zárójelezés nem bontható fel a kiszámítás gyorsítása érdekében!)

A. a+b*c+d*e

B. a*b+c*(d*e+f/g+h*i)

C. a*(b+c)-d*(e/f-g*h)+i*j

5. feladat: Pascal-program struktúrája (20 pont)

Egy Pascal-program struktúráját PROLOG-szerű *tényállításokkal* adjuk meg:

- `tartalmazza(KülsőEljárás, BelsőEljárást)`
- `megelőzi(EgyikEljárás, MásikEljárást)`

E tényállítások a közvetlen tartalmazást, illetve közvetlen megelőzést írják le.

Példa:

Itt van egy Pascal-program váza:

```
Program ...;
  Procedure első;
     Procedure második;
     end {második};
  end {első};
  Procedure harmadik;
  end {harmadik};
end {program}.
```

Struktúráját az alábbi tényállítások írják le:

```
tartalmazza("első","második").
megelőzi("első","harmadik").
```

A. Add meg a következő szabályoknak megfelelő Pascal-program vázát!

```
tartalmazza("A","B").
tartalmazza("B","C")
megelőzi("B","D").
megelőzi("A","E").
tartalmazza("A","D").
```

B. A Pascal-nyelvben, mint közismert, az eljárások a sorrendi és a láthatósági szabályok betartásával hívhatják egymást. Írj

```
hívhatja(X,Y) ha ...
```

néven olyan PROLOG-szerű eljárást, amely ellenőrzi, hogy `X` a sorrendi és láthatósági szabályok szerint hívhatja-e `Y`-t! A `ha` alapszó utáni formulákban a `nem`, a `vagy`, valamint az `és` logikai műveleteket, továbbá a tényállításokat is használhatod az általad esetleg definiálandó eljárások mellett.

6. feladat: Multihalmazok (24 pont)

Két állattenyésztő nyilvántartást vezet állatairól. Ez a nyilvántartás olyan `[` és `]` között álló, kétszeres mélységű sorozat, amely ábécé szerint rendezve tartalmazza az állatfajták nevét és darabszámát.

Példa:

```
A: [[kacsa,5],[liba,13], [nyúl, 66],[tyúk,4]]
B: [[kacsa,1],[kecske,33],[nyúl,99]]
```

A nyilvántartás kezelésére szolgáló függvények a következők:

`pár(E,M)`	`E`-ből és `M`-ből kételemű sorozatot hoz létre
`min(U,V)`	`U` és `V` közül a kisebbik
`első(X)`	az `X` sorozat első eleme
`második(X)`	az `X` sorozat második eleme
`elsőutániak(X)`	az `X` sorozat, az első eleme nélkül
`elejére(X,E)`	az `X` sorozatból és az elé illesztett `E` elemből álló sorozat

A. Mit adnak eredményül a következő rekurzív függvények?

```
Egyik(A,B):
  Ha üres(A) és üres(B) akkor 0
  különben ha üres(A) akkor második(első(B))+Egyik(A,elsőutániak(B))
  különben második(első(A))+Egyik(elsőutániak(A),B)
Függvény vége.
```

```
Másik(A,B):
  Ha üres(A) és üres(B) akkor []
  különben ha üres(A) akkor B
  különben ha üres(B) akkor A
  különben ha első(első(A))>első(első(B)) akkor
         elejére(első(B),Másik(A,elsőutániak(B)))
  különben ha első(első(A))<első(első(B)) akkor
         elejére(első(A),Másik(elsőutániak(A),B))
  különben elejére(pár(első(első(A)),második(első(A))+második(első(B))),
                   Másik(elsőutániak(A),elsőutániak(B)))
Függvény vége.
```

B. Készíts olyan függvényt, amely megadja, hogy a két gazdának összesen hány fajta állata van! (A fenti példában öt fajta állat szerepel.)

C. Készíts olyan függvényt, amely megadja, hogy melyik állatfajtából mennyit tudnak eladni akkor, ha egy-egy állatfajtából mindkettőjüknek ugyanannyit kell értékesíteniük! (A fenti példa szerint mindössze kacsát és nyulat adhatnak el, mégpedig:

```
[[kacsa,1], [nyúl,66]].)
```

7. feladat: Pénzgyűjtögetők (13 pont)

Arany Oszkár 5, Ezüst Benő 10 forintosokat gyűjt. Egy perselybe 5, 10 és 20 forintosokat lehet bedobni. Ha egymás után két 5 forintost dobnak be, akkor Arany Oszkár azonnal kicseréli őket egy

10 forintosra, ha pedig két 10 forintost dobnak be, akkor Ezüst Benő cseréli ki őket egy 20 forintosra.

Arany Oszkárt és Ezüst Benőt egy-egy olyan (végtelen ciklusban futó) eljárással helyettesítjük, amelyek egymást hívogatják. Az A eljárás végrehajtása félbeszakad, amikor a B-t meghívja, majd amikor a B hívja meg az A-t, akkor a B végrehajtása szakad félbe és az A végrehajtása folytatódik, amíg újra meg nem hívja a B-t. Ilyenkor az A megint félbeszakad, és a B végrehajtása folytatódik s.í.t. (A két eljárás végrehajtási módja tehát nem azonos a szokásos hívás-visszatérés mechanizmussal – az így működő eljáráspárt hívják korutinnak). `PÉNZ` a két eljárás közös változója.

```
Arany Oszkár:
  Ciklus
    Be: PÉNZ
    Ha PÉNZ=5
       akkor Be: PÉNZ
             Ha PÉNZ=5 akkor Ki: 10 (**)
                    különben P:=PÉNZ
                             PÉNZ:=5; Ezüst Benő
                             PÉNZ:=P; Ezüst Benő
             Elágazás vége
    különben Ezüst Benő
  Ciklus vége
Eljárás vége.

Ezüst Benő:
  Ciklus
    Ha PÉNZ=10 akkor Arany Oszkár   (***)
                     Ha PÉNZ=10 akkor Ki: 20 (*)
                             különben Ki: 10,PÉNZ
                     Elágazás vége
    különben Ki: PÉNZ
    Arany Oszkár
  Ciklus cége
Eljárás vége.
```

A. Milyen esetben hajtjuk végre a *-gal jelölt kiírást?

B. Ha Arany Oszkár kicserél két 5 forintost egy 10 forintosra, a fenti eljárás azonnal kiírja (**-gal jelölt sor). Alakítsd át úgy ezt a sort, hogy ha ezután 10 forintos következne, akkor Ezüst Benő beválthassa a két 10 forintost egy 20 forintosra!

C. Miért lenne helytelen az eljáráspár működése, ha a ***-gal jelölt helyen Arany Oszkár hívása helyett `Be: PÉNZ` állna?

1996. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Sportverseny (20 pont)

Egy kosárlabda csapatnak 5, egy kézilabda csapatnak pedig 7 tagja van. Egy osztály N (≥5) tanulóját úgy szeretnénk csapatokra osztani, hogy senki se szerepeljen egynél több csapatban, s a lehető legkevesebben maradjanak ki (például egy 18 fős osztályból 2 kosárlabda és 1 kézilabda csapatot szervezünk, mert így csak egyetlen tanuló marad ki, 19 fős osztály esetén azonban már 2 kézilabda és 1 kosárlabda csapatot kell szervezni).

Készíts programot, amely beolvassa egy osztály létszámát, majd kiírja, hogy belőlük hány kosárlabda és hány kézilabda csapatot szervezzünk, valamint, hogy hányan maradnak ki a csapatokból! (Ha több megoldás lenne, akkor is elég egyet - bármelyiket - megadni.)

2. feladat: Hét törpe (28 pont)

A hét törpe neve: Hapci, Kuka, Morgó, Szundi, Szende, Tudor és Vidor. Készíts programot, amely törpeneveket olvas be tetszőleges sorrendben, egyeseket akár többször is! Ha név begépelése helyett csak az ENTER billentyűt nyomjuk le, akkor kiírja, hogy hány különböző nevét gépeltünk be, és azt is, hogy melyiket hányszor (tetszőleges sorrendben, tetszőleges formában)!

3. feladat: Órák angolul (27 pont)

Az angol számnevek 1-től 12-ig: *one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve*. Két óra angolul: *two o'clock*. Negyed három angolul: *a quarter past two*. Fél öt angolul: *half past four*. Háromnegyed öt angolul: *a quarter to five*. Délelőtti órák (1.00-tól 11.45-ig) után az *a.m.*, délutániak (12.00-tól 23.45-ig) után pedig a *p.m.* rövidítést kell kitenni.

Készíts programot, amely beolvas két egész számot (az első 1 és 23 közötti egész szám jelenti az órát, a második 0, 15, 30 vagy 45 a percet), majd angolul kiírja a pontos időt!

Példa:

```
Bemenet: 21, 15
Eredmény: a quarter past nine p.m.
```

Kilencedik-tizedik osztályosok

1. feladat: Bűvös négyzet (18 pont)

A bűvös négyzet egy olyan N*N-es egész tömb, amelyben az elemek összege soronként, oszloponként és a két átló mentén egyenlő, továbbá egy-egy soron, ill. oszlopon belül ugyanaz a szám kétszer nem szerepel. Készíts programot, amely bűvös négyzetek ellenőrzésére használható! A BUVOSx.BE állomány első sora N értékét (≤10), valamint a várt sor-, oszlop- és átlóösszeget, a következő N sor pedig a tömb soraiban levő számokat tartalmazza, szóközökkel elválasztva.

Példa:

```
3 6
1 3 2
3 2 1
2 1 3
```

Az 1. átló az 1. sor 1. oszlopától a 3. sor 3. oszlopáig, a 2. átló pedig az 1. sor 3. oszlopától a 3. sor 1. oszlopáig tart.

1. sor, 1. oszlop	1. sor, 2. oszlop	1. sor, 3. oszlop
2. sor, 1. oszlop	2. sor, 2. oszlop	2. sor, 3. oszlop
3. sor, 1. oszlop	3. sor, 2. oszlop	3. sor, 3. oszlop

A program írja ki az IGEN szót, ha a bűvös négyzet helyes, illetve a NEM szót, ha hibás! Az utóbbi esetben ki kell írni a hibás összeget tartalmazó sorok, oszlopok, illetve átlók sorszámát a hibás összeggel együtt, továbbá azoknak a soroknak és oszlopoknak a sorszámát, amelyekben van ismétlődő érték.

2. feladat: Névnapok (24 pont)

A NEVNAPx.BE állomány (naptár) **legfeljebb** 365 sorból áll, K-adik sora az év K-adik napjára eső névnapokat tartalmazza. Ha egy napra csak egyetlen névnap esik, akkor a sorban egyetlen név

van; ha több, akkor a soron belül az egyes neveket egy-egy szóköz választja el. Ugyanazon a napon 5-nél több névnap nem lehet.

Készíts programot, amely

A. megadja, hogy hány név szerepel a naptárban;

B. beolvas egy nevet a billentyűzetről, s kiírja, hogy az adott név az év hányadik napján fordul elő a naptárban (ha többször is előfordul, az összeset megadja)!

3. feladat: Alakzatok (18 pont)

Síkbeli alakzatok egy csoportját a rájuk jellemző adatokkal együtt adjuk meg:

1. négyzet (oldalhossz),
2. kör (sugár),
3. egyenlő oldalú háromszög (oldalhossz).

Az ALAKx.BE állomány első sorában a feldolgozandó alakzatok száma (N), a következő N sorban pedig egy-egy alakzat (fentiek szerinti) sorszáma és jellemző adata van szóközzel elválasztva.

Készíts programot, amely megadja, hogy az alakzatok a felsorolás sorrendjében egymásba helyezhetők-e (úgy, hogy az első van legkívül, az utolsó pedig legbelül)! Ha igen, akkor a program az IGEN szót írja ki képernyőre! Ha nem, akkor a NEM szó után azoknak az alakzatoknak a sorszámát írja ki egy-egy szóközzel elválasztva, amelyekbe nem fér bele a soron következő alakzat!

4. feladat: Sportverseny (15 pont)

Egy kosárlabda csapat 5, egy kézilabda csapat 7, egy labdarúgó csapat pedig 11 tagú. Egy osztály N (≥5) tanulóját úgy szeretnénk csapatokra osztani, hogy senki se szerepeljen egynél több csapatban, s a lehető legkevesebben maradjanak ki (például egy 18 fős osztályból 1 labdarúgó- és 1 kézilabda csapatot szervezünk, mert így senki nem marad ki, 19 fős osztály esetén azonban már 2 kézi- és 1 kosárlabda csapatot kell szervezni).

Készíts programot, amely billentyűzetről beolvassa egy osztály létszámát, majd kiírja, hogy belőlük hány kosárlabda, hány kézilabda és hány labdarúgó csapatot szervezzünk, s hányan maradnak ki a csapatokból! Ha több megoldás lenne, akkor azt kell megadni, amelyikben a csapatok száma maximális (ezek szerint 35 tanulóból 7 kosárlabda csapatot kell szervezni, nem pedig 5 kézilabda csapatot).

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Sportverseny (12 pont)

Egy kosárlabda csapat 5, egy kézilabda csapat 7, egy labdarúgó csapat pedig 11 tagú. Egy osztály N (1 000 000 000≥N≥5) tanulóját úgy szeretnénk csapatokra osztani, hogy senki se szerepeljen egynél több csapatban, s a lehető legkevesebben maradjanak ki (például egy 18 fős osztályból 1 labdarúgó és 1 kézilabda csapatot szervezünk, mert így senki nem marad ki, 19 fős osztály esetén azonban már 2 kézi és 1 kosárlabda csapatot kell szervezni).

Készíts programot, amely billentyűzetről beolvassa egy osztály létszámát, majd kiírja, hogy belőlük hány kosárlabda, hány kézilabda és hány labdarúgó csapatot szervezzünk, s hányan maradnak ki a csapatokból! Ha több megoldás lenne, akkor azt kell megadni, amelyikben a csapatok száma maximális (ezek szerint 35 tanulóból 7 kosárlabda csapatot kell szervezni, nem pedig 5 kézilabda csapatot).

2. feladat: Kerítésfestés (19 pont)

Sebaj Tóbiás házát léckerítéssel vette körül, s be szeretné festeni. Különböző színű festékei vannak. Mindegyikről tudja, hogy hány méter kerítést lehet vele befesteni. Tóbiás nagyon takarékos, ezért ha egyszer egy festékesdobozt megkezd, el is használja.

Segíts neki, készíts olyan programot, amely megmondja, hogy mely festékeket kell használnia! Ha több lehetőség is van, akkor a programnak azt kell megadnia, amelyikhez a legkevesebb festékesdobozt kell kinyitnia.

A KERITESx.BE állomány tartalma a következő:

- első sor: a kerítés hossza (H),
- második sor: a festékesdobozok száma (N),
- következő N sor: a dobozokban levő festékkel befesthető kerítésszakasz hossza (egész számként), s tőle egyetlen szóközzel elválasztva a festék színe.

A KERITESx.KI állományba, valamint a képernyő első sorába a „BE LEHET FESTENI”, vagy a „NEM LEHET BEFESTENI” szöveget kell írni. Ha be lehet festeni a kerítést a megadott feltételek mellett, akkor a következő sorba a felhasznált dobozok számát (M), a következő M sorba pedig a felhasznált dobozokban levő festék színét és a vele befesthető kerítésszakasz hosszát kell írni, az utóbbiakat egyetlen szóközzel elválasztva. Ha nem lehet befesteni a kerítést, akkor vagy a „NINCS ELÉG FESTÉK”, vagy a „MARAD VALAMELYIK FESTÉKBŐL” szöveget kell írni a második sorba.

3. feladat: Autóverseny (26 pont)

Egy Forma-1-es versenypálya alakját egy betűsorozattal írjuk le, amelyben a B,E,J betűk szerepelhetnek. Az egyes betűk jelentése:

B: balkanyar E: egyenes szakasz J: jobbkanyar

Mindhárom betű S méter hosszúságú útszakaszt jelöl. Az autók 0 m/s sebességgel indulnak a rajthelyről, egyenes szakaszon a maximális sebességük M m/s, a gyorsulásuk pedig állandó, maximum G m/s^2. Kanyarban az autók maximális sebessége K m/s, gyorsítaniuk, illetve lassítaniuk nem szabad.

Emlékeztetőül:

A T idő alatt megtett út: $S = V * T + 1 / 2 * G * T^2$

G gyorsulás hatására a sebesség T idő múlva: $V_{új} = V + G * T$

Készíts programot, amely egy adott versenypálya minden szakaszára kiszámolja, hogy a pályán a lehető leggyorsabban haladó autó az egyes szakaszok végén milyen sebességet ér el, és meghatározza a pálya megtételéhez szükséges időt!

S, M, G és K értékét a program a billentyűzetről olvassa be!

A FORMA1x.BE állomány minden egyes sorában egy-egy versenypályát leíró betűsorozat van. A FORMA1x.KI állományba ugyanennyi sort kell írni; az egyes sorokban a pályaszakaszok végén elért sebességértékeket kell felsorolni, végül a pálya megtételéhez szükséges időt kell megadni. A egyes számokat legalább egy szóközzel kell elválasztani egymástól.

4. feladat: Nyomtatás (18 pont)

Az ISKOLAx.BE állományban egy város középiskolás tanulóiról az alábbi adatokat tároljuk (minden adat szöveg típusú, minden adat külön sorban van):

1. A tanuló neve

2. A tanuló születési ideje
3. A középiskola neve
4. A tanuló évfolyama
5. A tanuló osztálya
6. A tanuló tanulmányi átlaga

Az állomány a (3, 4, 5, 1) sorszámú pontok alapján lexikografikusan rendezve van.

Írj programot, amely kilistázza a tárolt adatok közül a tanulók *nevét, születési idejét* és *tanulmányi átlagát* három egymás utáni sorban. Egy tanuló minden adatának ugyanarra az oldalra kell kerülnie.

Az eredményt nyomtató helyett az ISKOLAx.KI állományba és a képernyőre kell írni. A listát oldalakra kell bontani (az állományban az oldal végét egy legalább 40 db – jelből álló sor jelezze, képernyőn pedig minden lap végén billentyűlenyomásra kell várni a programnak, majd képernyő törlés után folytatni kell a kiírást). Egy oldalra *N* db sor fér (plusz a fejléc és a lábléc). Minden oldal tetejére fejlécet, aljára (a láblécbe) a lap sorszámát (1-től kezdve) kell írni. A lapra írható sorok *N* számát, valamint az egysoros fejlécszöveget a program a billentyűzetről olvassa be.

Egy-egy iskola, évfolyam vagy osztály adatai elé ún. bevezető sort kell írni az alábbiak közül a megfelelő szöveggel:

- Az XY középiskola adatai
- Az *XY* középiskola *n.* évfolyamának adatai
- Az *XY* középiskola *n.* évfolyama *k.* osztályának adatai

Az iskolákhoz tartozó bevezető sorok külön oldalon legyenek. Az évfolyamokhoz tartozó bevezető sorok új oldal tetejére kerüljenek. Az osztályok bevezető sorai előtt 2–2 üres sort kell hagyni (kivéve, ha lap tetejére kerülnek). Az évfolyamok és az osztályok bevezető sorát egy-egy üres sor kövesse.

Az iskola, az évfolyam és az osztály adatainak kiírása után egy üres sor (a lap teteje kivételével), majd egy összefoglaló sor következzék. Az összefoglaló sor tartalmazza

- iskola esetén az évfolyamok, osztályok, illetve tanulók számát;
- évfolyam esetén a legfiatalabb tanuló születési idejét;
- osztály esetén az osztály tanulmányi eredményeinek átlagát.

A listázás során arra is ügyelni kell, hogy egy-egy osztály bevezető sora és legalább egy tanulójának adatai ugyanazon az oldalon legyenek.

1996. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Tükör-telefonszámok (16 pont)

A budapesti telefonszámok hétjegyűek, a vidékiek a kétjegyű körzetszámmal együtt nyolcjegyűek.

Példa: egy budapesti telefonszám — 1234567, egy siófoki — 84123456, ebből az első két számjegy a körzetszám, a következő hat számjegy pedig a körzeten belüli, helyi telefonszám. Készíts programot, amely beolvas egy egész számot (N), majd pedig N telefonszámot, s kiírja azokat, amelyek vagy ugyanazt a számot adják megfordítva (pl. 1234321 vagy 84122148), vagy a körzetszámuk ugyanaz megfordítva (pl. 66123456), vagy a körzetszám nélküli részük (72123321) ugyanaz megfordítva!

2. feladat: Névsor (31 pont)

Készíts programot, amely beolvassa egy osztály tanulóinak nevét, majd megadja, hogy névsorban vannak-e!

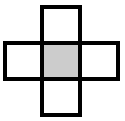
Ha névsorban vannak, akkor azt a szöveget írja ki, hogy a NÉVSOR JÓ, továbbá azt, hogy növekvő vagy csökkenő sorrendben vannak-e a nevek.

Ha nincsenek névsorban, akkor eldönti, hogy a sorrend inkább növekvő vagy inkább csökkenő-e! Inkább növekvő az a sorrend, amelyben növekvő sorrendű szomszédos párból több van, mint csökkenőből, de legfeljebb velük megegyező számú. Inkább csökkenő a sorrend, ha csökkenő sorrendű szomszédos párból van több. (Pl. az AA DD CC BB GG FF csökkenő sorrendű sorozat, mert a szomszédos párok közül 3 van csökkenő sorrendben, s csak 2 növekvőben). Ezután kiírja azokat az egymás után álló neveket, amelyek az így megállapított sorrend szerint rossz sorrendben vannak. (Az előző példa alapján az AA DD, illetve a BB GG párokat kell kiírni.)

A program először a tanulók számát (N) olvassa be, majd az összes tanulók nevét, s csak ezután írja ki az eredményt! A nevekben a magyar ábécé kis- és nagybetűi, valamint szóközök lehetnek, a magyar nevek helyesírási szabályának megfelelően.

A magyar ábécé ékezetes betűinek kódja: á=160, é=130, í=161, ó=162, ö=148, ő=147, ú=163, ü=129, ű=150, Á=143, É=144, Í=141, Ó=149, Ö=153, Ő=167, Ú=151, Ü=154, Ű=152.

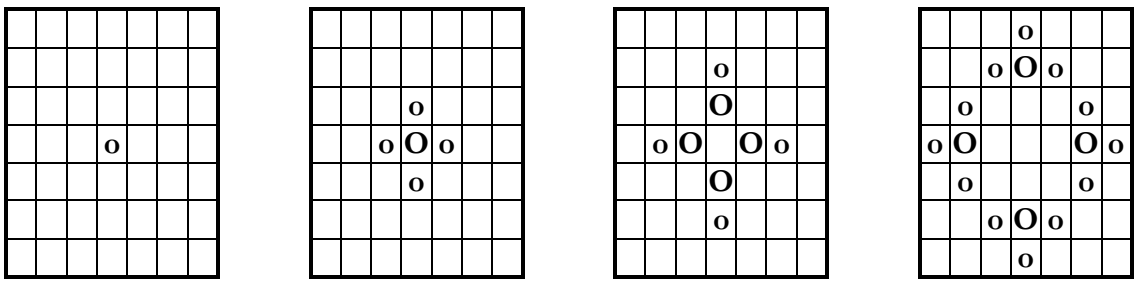
3. feladat: Szimmetriajáték (23 pont)



Stanislaw Ulam szimmetriajátéka négyzetrács alakú táblán játszódik. Kezdetben a négyzetrács 1 vagy 2 pontján van kis **o** betű. Ezután minden következő lépésben:

- minden olyan üres helyen, amelynek „oldalszomszédságában” (a jobbra levő ábrán a satírozott mező „oldalszomszédai” látszanak) egyetlen kis **o** vagy nagy **O** betű található, megjelenik egy kis **o** betű;
- minden eddigi kis **o** betűből nagy **O** betű lesz;
- minden eddigi nagy **O** betű eltűnik a tábláról.

Példa:



kezdőállapot 1. lépés 2. lépés 3. lépés

Készíts programot, amely megkérdezi, hogy 1 vagy 2 betűt rakjon-e kezdetben a 25*25-ös méretű induló táblára, majd beolvassa a helyek koordinátáit. Ezután a kért helyekre elhelyezi a kis **o** betűket, majd a kezdőállapotból tetszőleges billentyű lenyomására egymás után előállítja és kiírja a képernyőre a soronkövetkező állapotokat!

Kilencedik-tizedik osztályosok

1. feladat: Logo (75 pont)

A LOGO programozási nyelv utasításai közül az alábbiakat használjuk:

Mozgások:

FORWARD x, BACK x — az aktuális irányba előre, illetve hátra lép x egységgel;

Forgások:

LEFT x, RIGHT x	—	az aktuális iránytól balra, illetve jobbra fordul x fokot; közben nem rajzol

Toll helyzet állítások:

UP, DOWN	—	felemeli, illetve leteszi a tollat, felemelt tollal haladás közben nem rajzol, kezdetben a toll a papíron van.

Az utasítások x és y paramétere lehet pozitív vagy negatív egész állandó.

Készíts olyan kódoptimalizáló programot, amely a következő átalakításokra képes:

FORWARD x FORWARD y	→	FORWARD x+y	(azonos előjelű x és y, vagy felemelt toll esetén)
BACK x BACK y	→	BACK x+y	(azonos előjelű x és y, vagy felemelt toll esetén)
FORWARD x BACK y	→	FORWARD x–y	(ellenkező előjelű x és y, vagy felemelt toll esetén)
BACK x FORWARD y	→	BACK x–y	(ellenkező előjelű x és y, vagy felemelt toll esetén)
LEFT x LEFT y	→	LEFT x+y	
RIGHT x RIGHT y	→	RIGHT x+y	
LEFT x RIGHT y	→	LEFT x–y	
RIGHT x LEFT y	→	RIGHT x–y	
DOWN forgások UP	→	forgások	(kezdetben felemelt toll esetén)
UP forgások DOWN	→	forgások	(kezdetben leengedett toll esetén)
DOWN mozgás, forgás DOWN	→	DOWN mozgás, forgás	
UP mozgás, forgás UP	→	UP mozgás, forgás	

Az átalakítások közben a műveleteket el is kell végezni (LEFT 30+30 helyett LEFT 60)!

Az összességében 0 fokos fordulatokat és a 0 elmozdulásokat (ha nincs hatásuk) törölni kell!

Az átalakítandó LOGO program a LOGOx.BE állományban található, sorvégekkel, szóközökkel tetszőlegesen tördelve. Az eredményt a LOGOx.KI állományba kell írni, minden utasítást külön sorba, paraméterétől egyetlen szóközzel elválasztva!

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Televízióadások (55 pont)

A TV-készülékeden egyszerre N televízió-csatornát foghatsz. A szilveszteri műsort mindegyik adó igyekszik jó filmekkel kitölteni. Előfordulhat, hogy nem tudod mind megnézni. Készíts programot, amely az alábbi szempontok szerint válogatja ki a filmeket:

- A lehető legtöbb filmet szeretnéd megnézni.
- Csak teljes filmeket nézel meg (azaz ha akár egyetlen percet elmulasztasz egy filmből, akkor azt már nem nézed meg).
- Egyszerre csak egy filmet nézhetsz a TV-készülékeden.

A számodra érdekes filmeket (amelyek közül a program válogat) a TVx.BE állományban tároljuk. Az állomány soronként egy film leírását tartalmazza, a soron belül az adatokat egy-egy szóköz választja el. A film leírása a következő elemekből áll:

`adó sorszáma, kezdési idő (óra, perc), befejezési idő (óra, perc)`

Az állomány sorairól semmiféle rendezettség nem tehető fel.

Készíts menürendszerű programot az alábbi funkciók végrehajtására:

A. Az összes film kiírása a TVx.KIA állományba és a képernyőre TV-csatornák, s ezen belül kezdési idő szerinti sorrendben!

B. A lehető legtöbb film adatai felsorolása (a TVx.KIB állományban és a képernyőn) kezdési idő szerinti sorrendben, amit megnézhetsz. Ha több megoldás is van, akkor azt kell megadni, amelynek az összideje a legnagyobb. (Törekedj arra, hogy a program e része optimális futási idejű legyen!)

C. A lehető legtöbb film adatai felsorolása (a TVx.KIC állományban és a képernyőn) kezdési idő szerinti sorrendben, amit megnézhetsz vagy felvehetsz videóra, ha van egy videomagnód, amelyre a nézett filmmel párhuzamosan felvehetsz egy másik csatornán sugárzott filmet.

Egy-egy film adatait követően írd ki a TV szót, ha azonnal megnézed a filmet, illetve a VIDEÓ szót, ha felveszed későbbi megnézés céljából.

D. A lehető legtöbb film adatai felsorolása (a TVx.KID állományban és a képernyőn) kezdési idő szerinti sorrendben, amit megnézhetsz, vagy felveheted videóra, ha van egy videómagnód, amelyre a nézett filmmel párhuzamosan felvehetsz egy másik csatornán sugárzott filmet, de csak egyetlen tetszőleges hosszúságú filmet vehetsz fel, s az újabb film felvétele előtt a korábban felvettet meg kell nézned. Éjfélig (24 óra 0 percig) az utolsó videokazettára felvett filmet is meg kell nézned.

Egy-egy film adatait követően írd ki a TV szót, ha azonnal megnézed a filmet, a VIDEÓFELVÉTEL szót, ha felveszed későbbi megnézés céljából, illetve a VIDEÓLEJÁTSZÁS szót, ha a kazettán levő filmet nézed (az előtte levő számok ilyenkor a lejátszás időpontjait jelentik).

E. Befejezés.

2. feladat: Maga mindent kétszer mond, kétszer mond? (20 pont)

Az `X→Y` jelöléssel *(szabállyal)* azt fejezzük ki, hogy az `X` halmazba tartozó értékek egyértelműen meghatározzák az `Y` halmazba tartozó értékeket.

Például a személyi szám (`S`) meghatározza egy személy nevét (`N`), címét (`C`), telefonszámát (`T`) és születési helyét (`H`), azaz `S→N`, `S→C`, `S→T` és `S→H`, vagy tömörebben `S→NCTH`. Az `NC→T` szabály azt jelenti, hogy egy személy neve és címe alapján megtudhatjuk a telefonszámát. Az `S→N`, `S→C` és `NC→T` szabályok mellett az `S→T` szabály nem mond újat, hiszen a személyi számból meghatározható a név és a cím, e kettő alapján pedig a telefonszám, ezért az `S→T` szabályt felesleges megadni.

Írj programot, amely a `KMKMx.BE` szöveges állományból

1. beolvassa szabályok egy halmazát, amelyben nincs felesleges szabály,
2. beolvas egy újabb szabályt, és kiírja a `KMKMx.KI` állományba, hogy ez a szabály felesleges-e, azaz levezethető-e az (1) lépésben megadott szabályokból,
3. a (2) lépést addig ismétli, amíg újabb szabályhalmazra nem bukkan,
4. az (1), (2) és (3) lépéseket addig ismétli, amíg van mit olvasnia a bemeneti állományból.

A `KMKMx.BE` bemeneti állomány szerkezete:

A szöveges állományban tetszőleges számú, szintaktikailag helyes szabálycsoport van. Egy-egy szabálycsoport három részből áll: egy egész számból, egy szabályhalmazból és további szabályokból. A szám 0-99 közötti érték lehet, és azt mondja meg, hogy a szabályhalmazban hány szabály van. A

szabályhalmaz után tetszőleges számú további szabály következhet, ezek felesleges voltát kell megállapítania a programnak. Minden szabálynak egy vagy több nagybetűből álló bal és jobb oldala van, a két oldalt a `->` jelpár választja el egymástól. A szabálycsoport kezdetét jelző egész szám és minden egyes szabály is új sorban van. Mivel szabályok csak nagybetűvel kezdődhetnek, újabb szabálycsoport kezdetét a szabályhalmaz elemeinek számát megadó egész szám jelzi; ha ez 0, a bemeneti állománynak vége van. A bemeneti állományban nincs szóköz, sem üres sor.

A `KMKMx.KI` bemeneti állomány szerkezete:

A kimeneti állományba és a képernyőre át kell másolni az összes olyan szabályt a bemeneti állományból (az előfordulásuk sorrendjében), amelyek feleslegesek. Minden sorba egy szabályt és utána a `felesleges` szót, ha nincs felesleges szabály, akkor a `Semmi sem felesleges` szöveget kell írni. Az eredetileg külön szabálycsoportokba tartozó szabályokat a `Szabálycsoport #`*n* szöveget tartalmazó sorral kell bevezetni, ahol *n* a szabálycsoport sorszáma (1-től kezdve, folyamatos sorszámozással).

A verseny végeredménye:

I. kategória

1. Nepusz Tamás	Teleki Blanka Gimnázium, Székesfehérvár
2. Vágó Dávid	Radnóti Miklós Gimnázium, Budapest
3. Salvi Péter	Eötvös József Gimnázium, Budapest
3. Kasza Péter	Fazekas Mihály Gimnázium, Debrecen
5. Varga Kornél	Jókai Mór Általános Iskola, Sátoraljaújhely
Herendi Gergely	Radnóti Miklós Gimnázium, Dunakeszi
7. Hegedűs Ramon	Bolyai Általános Iskola és Gimnázium, Szombathely
8. Rokob András	Szilágyi Dezső Általános Iskola, Miskolc
Ivaskó György	III. Béla Gimnázium, Baja
10. Oláh István	Krúdy Gyula Gimnázium, Nyíregyháza

1996. II. kategória

1. Nagy András	Leőwey Klára Gimnázium, Pécs
2. Terék Zsolt	Fazekas Mihály Gimnázium, Budapest
3. Várady Gergő	Eötvös József Gimnázium, Budapest
4. Radnai Zoltán	Alternatív Közgazdazdasági Gimnázium, Budapest
5. Várkonyi Dániel	Teleki Blanka Gimnázium, Székesfehérvár
6. Förhécz András	Teleki Blanka Gimnázium, Székesfehérvár
7. Felföldi Zsolt	Fazekas Mihály Gimnázium, Budapest
8. Czimmermann Péter	Fazekas Mihály Gimnázium, Budapest
9. Csíkvári András	Fazekas Mihály Gimnázium, Budapest
10. Hartmann Miklós	Petőfi Sándor Evangélikus Gimnázium, Bonyhád

1996. III. kategória

1. Noll János	Fazekas Mihály Gimnázium, Budapest
2. Tóth László	Földes Ferenc Gimnázium, Miskolc

3. Kovács Gábor Zsolt	Lovassy László Gimnázium, Veszprém
4. Ungár Péter	Fazekas Mihály Gimnázium, Budapest
5. Klampeczki Zsolt	Gépészeti és Számítástechnikai Szakközépiskola, Békéscsaba
Peller Balázs	Neumann János Szakközépiskola, Budapest
7. Zsila Zsolt	Neumann János Szakközépiskola, Budapest
8. Tolvaj Béla	Földes Ferenc Gimnázium, Miskolc
Kurucsai Tamás	Óbudai Gimnázium, Budapest
10. Lakatos Roland	Zrínyi Miklós Gimnázium, Zalaegerszeg
Hock Szabolcs	Kőrösi Csoma Sándor Gimnázium, Hajdúnánás

1997. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendszámok (20 pont)

Néhány évvel ezelőtt az autók rendszáma két betűből és négy számjegyből állt (pl. ZZ1234). Készítettek egy programot, amely véletlenszerűen kiválaszt egy rendszámot. A véletlenbetű("Z") függvényhívás egy A és Z közötti betűt ad véletlenszerűen, a véletlenszámjegy(9) pedig egy 0 és 9 közötti számjegyet.

```
Rendszámválasztás:
  Ciklus I=1-től 2-ig
     REND(I):=Véletlenbetű("Z")
  Ciklus vége
  Ciklus I=3-tól 6-ig
     REND(I):=Véletlenszámjegy(9)
  Ciklus vége
Eljárás vége.
```

Jelenleg a rendszámok egy jellel hosszabbak, három betűből, egy kötőjelből és három számjegyből állnak (pl. ABC-123). Alakítsd át a fenti programot úgy, hogy újfajta rendszámokat állítson elő, azzal a megkötéssel, hogy az utolsó érvényes rendszám a GZZ-999!

2. feladat: Rajzoló (42 pont)

A LOGO nyelv alábbi utasításait használjuk:

`forward h, back h`	– az aktuális irányba előre-, ill. hátra lép `h` egységgel
`left sz, right sz`	– az aktuális helyen elfordul balra, ill. jobbra `sz` fokkal
`REPEAT db [utasítások]`	– a zárójelbe zárt utasításokat db-szer megismétli
`IF` feltétel `THEN` utasítások	– a THEN mögötti, valamint a következő sorokban bekezdéssel írt utasításokat végrehajtja, ha az IF mögötti feltétel teljesül

Mit rajzolnak az alábbi LOGO programok a képernyőre (n=1, h=100), (n=2, h=100), illetve (n=3, h=100) esetén, ha a teknőc kezdetben észak felé néz? Rajzold le!

```
A. a(n,h):
     IF n>0 THEN REPEAT 3 [forward h left 90 a(n-1,h/3)
                           left 180 a(n-1,h/3) left 90]
                 back 3*h

B. b(n,h):
     IF n>0 THEN forward h/2
                  REPEAT 2 [forward h/2 left 90
                         b(n-1,h/2) right 90
                         forward h/2 right 90
                         b(n-1,h/2) left 90]
                 back 5*h/2

C. c(n,h):
     IF n>0 THEN REPEAT 4 [forward h/2 left 90
                           c(n-1,h/3) right 90]
                 back 2*h
```

3. feladat: Távolugrók (18 pont)

Feljegyeztük minden távolugró legjobb eredményét az idei olimpián, ezeket az X(1..N) tömb tartalmazza.

```
Valami:
  Ha X(1)>X(2) akkor  A:=X(1); B:=X(2)
              különben A:=X(2); B:=X(1)
  Ciklus I=1-től N-ig
     Ha X(I)>B akkor
        Ha X(I)>A akkor B:=A; A:=X(I) különben B:=X(I)
  Ciklus vége
Eljárás vége.
```

A. Mit csinál a fenti algoritmus? Mi lesz A és B értéke a futás végén? Mi a helyzet holtverseny esetén?

B. Milyen esetben eredményez(het) futási hibát a program futtatása?

C. Mire módosíthatók a ciklushatárok úgy, hogy az eredmény ne változzon?

4. feladat: Kerítésmagasságok (20 pont)

A rátóti körút lakosai újfajta versengésbe kezdtek. Azon mesterkednek, hogy a kerítésük magasabb legyen mindkét szomszédjuk kerítésénél. A körúton N ház van egymás mellett, a kerítésük magasságát az X(1..N) tömbben tároljuk. (A körút első és utolsó háza is szomszédos.)

```
Kerítés:
  A:=X(1); B:=X(2); L:=hamis
  Ciklus I=3-tól N-ig
     Ha B>X(I)és B>A akkor L:=igaz
     A:=B; B:=X(I)
  Ciklus vége
Eljárás vége.
```

A. Milyen esetben lesz igaz az `L` változó értéke a futás végén?

B. Mi a hiba a programban? Javítsd ki a lehető legegyszerűbb módon!

C. Mi az `A` és `B` változók szerepe?

Kilencedik-tizedik osztályosok

1. feladat: Halmazműveletek (10 pont)

Az A és B halmazok egész számokat tartalmaznak. Mindkét halmaznak van olyan eleme, amelyik a másiknak nem eleme. Használni fogjuk a következő utasításokat:

nemüres(H)	igaz, ha a H halmaznak legalább egy eleme van.
eleme(e, H)	igaz, ha az e elem benne van a H halmazban.
üreshalmaz	üres halmazt ad eredményül.
egyeleme(H)	a H halmaz egy tetszőleges elemét adja eredményül.
hozzáad(e,H)	az e elemet berakja a H halmazba.
kivesz(e,H)	az e elemet kiveszi a H halmazból.

```
Valami:
  H:=üreshalmaz
  Ciklus amíg nemüres(A) vagy nemüres(B)
     Ha nemüres(A) akkor e:=egyeleme(A) különben e:=egyeleme(B)
     Ha eleme(e,A) akkor kivesz(e,A)
     Ha eleme(e,B) akkor kivesz(e,B)
     hozzáad(e,H)
  Ciklus vége
Eljárás vége.
```

A. Mit csinál a program? Mi lesz a futás végén a H halmazban, A-ban és B-ben?

B. Hogyan változik a három halmaz tartalma, ha a ciklusfeltételben szereplő `vagy` művelet helyére `és` műveletet írunk?

2. feladat: Rajzolgatunk (16 pont)

Az alábbi LOGO programokban egyszerre 2 teknőc rajzol. Azonos kezdőállapotból indulnak, kelet felé, a tollak a papíron vannak. Rajzold le az általuk elkészített ábrákat! Add meg az egyes szögek nagyságát fokban, és nevezd meg a jellegzetes geometriai alakzatokat is!

```
A. Teknőc1:
     right 90 forward 10 left 90 forward 100 right 90
     forward 20 left 120 forward 60

   Teknőc2:
     left 90 forward 10 right 90 forward 100 left 90
     forward 20 right 120 forward 60

B. Teknőc1:
     right 135 forward négyzetgyök 200 left 135
     forward 100 right 120 forward 20 left 150
     forward 20+20*négyzetgyök 3

   Teknőc2:
     left 135 forward négyzetgyök 200 right 135
     forward 100 left 120 forward 20 right 150
     forward 20+20*négyzetgyök 3
```

3. feladat: Programstruktúrák kezdete és vége (18 pont)

Egy programozási nyelvben az elágazás (IF utasítás) végét mindig a FI, a ciklus (DO utasítás) végét pedig az OD alapszó jelzi. Készítettünk egy algoritmust, amely ellenőrzi az IF-FI, illetve a DO-OD párok helyes párosítását. A megoldásban az N elemű X vektor tartalmazza a program szavait.

```
Ellenőr:
  A:=0; B:=0
  Ciklus I=1-től N-ig
     Elágazás
        X(I)="IF" esetén A:=A+1; V(A):=B
        X(I)="DO" esetén B:=B+1
        X(I)="FI" esetén A:=A-1; Ha A<0 akkor HIBA1
                         Ha V(A+1)<B akkor HIBA2
                         Ha V(A+1)>B akkor HIBA3
        X(I)="OD" esetén B:=B-1; Ha B<0 akkor HIBA4
     Elágazás vége
  Ciklus vége
  Ha A>0 akkor HIBA5
  Ha B>0 akkor HIBA6
Eljárás vége.
```

Milyen hibajelenségek váltják ki a HIBA1..HIBA6 hibajelző utasításokat?

4. feladat: Hőmérsékletmérések (14 pont)

Kukutyinban N napon keresztül mértük a hőmérsékletet. Az egyes napi mérések értékét az A(1..N) tömb tartalmazza.

```
Valami:
  L:=hamis
  Ciklus I=1-től N-ig
     Ha A(I)<0 akkor
        Ha nem L akkor L:=igaz; P:=I; M:=A(I)
        különben ha A(I)>M akkor P:=I; M:=A(I)
     Elágazások vége
  Ciklus vége
Eljárás vége.
```

A. Mit csinál a fenti program?

B. Mi a jelentése az `L`, `P` és `M` változóknak?

5. feladat: Festőalgoritmus (18 pont)

A képernyőn levő zárt alakzatok befestésére készítettük az alábbi, nem mindig jól működő algoritmust: Az alakzatot olyan *festett* pontok határolják, amelyek 8 szomszédja között legfeljebb két határpont van.

```
Festés:
  Ciklus sor=felső-től alsó-ig
     fest:=hamis
     Ciklus oszlop=bal-tól jobb-ig
        Ha festett(oszlop,sor) akkor fest:=nem fest
        Ha fest akkor pontrajzolás(oszlop,sor)
     Ciklus vége
  Ciklus vége
Eljárás vége.
```

A. Mikor „gondolja” úgy a fenti festőalgoritmus, hogy zárt alakzat belsejében van?

B. A képet egyes esetekben nem jól festi. Melyek ezek?

6. feladat: Pincérek és vendégek (24 pont)

Egy étteremben, ahol K pincér dolgozik, N vendég ebédelhet egyszerre. A vendéget a `Rendelés`, a pincért a `Kiszolgálás` eljárás szimulálja. Egyszerre több vendég is rendelhet, és őket több pincér szolgálhatja ki. Amikor egy pincér befejezi egy vendég kiszolgálását, azonnal hozzáfoghat a következő kiszolgálásához.

A rendeléseket (az étel nevét és a vendég sorszámát) a REND(1..N) tömbben tároljuk. A `Rendelés` végrehajtása 2, a `Kiszolgálás` végrehajtása 5 percig tart.

Például, ha egy vendég a 3. percben kezdi el tanulmányozni az étlapot, rendelését legkorábban az 5. percben veszi fel egy pincér. Ha ekkor éppen van szabad pincér, a vendég pontosan a 10. percben kapja meg az ételt, ha nincs, akkor várnia kell. A pincér a kiszolgálás befejezésének percében (azaz a példa szerint a 10. percben) már elkezdheti a következő vendég kiszolgálását.

```
Rendelés(ÉTEL,VENDÉG):
  REND(I):=(ÉTEL,VENDÉG)
  I:=(I+1) mod N; DB:=DB+1
Eljárás vége.
```

```
Kiszolgálás(ÉTEL,VENDÉG):
  (ÉTEL,VENDÉG):=REND(J)
  J:=(J+1) mod N; DB:=DB-1
Eljárás vége.
```

A. Milyen programozási hiba van az algoritmusban? Javítsd ki! (Új utasítást nem írhatsz bele!)

B. Mi legyen az I, a J és a DB változók kezdőértéke?

C. Nézzük azt az esetet, amikor a vendégek a 6., 7., 9., 15. és 20. percben kezdik el nézegetni az étlapot, és az étteremben két pincér dolgozik. Melyik vendég mikor kapja meg az ételt a fenti algoritmus szerint?

D. A C. pontbeli adatok mellett mikor van mindkét pincérnek egyszerre munkája?

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rekurzió (20 pont)

A TÖRPE(1..5) tömbben a "HAPCI" szöveg található.

```
Valami(I):                        Print:
  Ciklus J=1-től 5-ig               Ciklus K=1-től 5-ig
     T(I):=J                           Kiír(TÖRPE(T(K)))
     Ha I=5 akkor Print             Ciklus vége
       különben Valami(I+1)         Sorvég-írás
  Ciklus vége                     Eljárás vége.
Eljárás vége.
```

A. Mit ír ki a program a Valami(1) eljáráshívás hatására? (A kiírás sorrendjét nem kell megadni!

B. Mit tartalmaz a T tömb?

C. A Valami eljárás hányszor hívja meg a Print eljárást?

D. Mit ír ki a program a Valami(1) eljáráshívás hatására, ha a Valami eljárásban a ciklus magját az alábbira cseréljük:

```
Ha J∉H akkor H:=H∪Hiba! A könyvjelző nem létezik.{J}; T[I]:=J
             Ha I=5 akkor Print különben Valami(I+1)
             H:=H-{J}
Elágazás vége
```

E. Hányszor hívja meg most a Valami eljárás a Print eljárást?

2. feladat: Közös adatbázis (18 pont)

Egy adatbázist hálózaton keresztül több program is használ. Ez azonban veszélyeket rejt magában, például az éppen olvasott adatot felülírhatja egy másik program. Az ilyen veszélyek elkerülésére ún. *primitív utasításokkal* szabályozzuk az adatbázis írását és olvasását. A **Küld, Van** és **Töröl** primitív utasítások jellegzetessége, hogy közülük egy időben csak egyet hajt végre az adatbázist felügyelő program. A várakozó primitív utasítások végrehajtására érkezésük sorrendjében kerül sor.

A **Küld(szöveg)** utasítás hatására a felügyelő program feljegyzi a kapott szöveget. A **Van(szöveg1, szöveg2,...)** függvény értéke igaz, ha a megadott szövegek közül legalább egyet tárol a felügyelő program. Tárolt szöveget a **Töröl(szöveg)** utasítással törölhetünk. Adatbázist író és olvasó programokból tetszőlegesen sok futhat egyszerre.

Részlet az író programokból:

```
Az írandó adatok elkészítése
Küld("Írási szándék")
Vár amíg Van("Olvasás","Írás")
Küld("Írás")
Töröl("Írási szándék")
Az adatbázis írása
Töröl("Írás")
```

Részlet az olvasó programokból:

```
Az olvasandó adatok meghatározása
Küld("Olvasási szándék")
Vár amíg Van("Írás")
Küld("Olvasás")
Töröl("Olvasási szándék")
Az adatbázis olvasása
Töröl("Olvasás")
Az olvasott adatok feldolgozása
```

A. Milyen feltételek mellett írhatják-olvashatják az adatbázist ezek a programrészletek?

B. Az író program(ok) futását az olvasó programok, ha több van belőlük, tetszőlegesen hosszú időre felfüggeszthetik. Hogyan?

C. Mit és hogyan kell megváltoztatni a fenti programrészletekben ahhoz, hogy a B. pontban említett jelenség ne következhessen be?

D. Ha a fenti író programok közül egynél több fut egyidejűleg, előfordulhat, hogy az adatbázist elrontják. Milyen esetben?

3. feladat: Telefonszámok (12 pont)

Négyjegyű telefonszámokat tárolunk az N elemű TEL vektorban, teljesen rendezetlenül. Az alábbi függvény egy telefonszámot ad eredményül:

```
Telefonszám(A,F):
  DB1:=0; DB2:=0
  Ciklus I=1-től N-ig
     Ha TEL(I)≥ A és TEL(I)≤(A+F) div 2 akkor DB1:=DB1+1
     Ha TEL(I)≤F és TEL(I)>(A+F) div 2 akkor DB2:=DB2+1
  Ciklus vége
  Ha DB1=0 akkor Telefonszám:=A
  különben Ha DB2=0 akkor Telefonszám:=F
  különben Ha DB1≤DB2 akkor Telefonszám:=Telefonszám(A,(A+F) div 2)
  különben Telefonszám:=Telefonszám((A+F) div 2+1,F)
Függvény vége.
```

A. Mi lesz a Telefonszám(1000,9999) hívás eredménye, ha a TEL vektorban 100 db 2-vel kezdődő telefonszám van?

B. Mi lesz a Telefonszám(1000,9999) hívás eredménye, ha a TEL vektor az összes 000-ra végződő telefonszámot tartalmazza, és más számot nem?

C. Milyen tulajdonságú értéket ad eredményül a Telefonszám függvény?

D. Legfeljebb hányszor hívhatja meg saját magát a Telefonszám függvény a Telefonszám(1000,9999) hívás hatására?

4. feladat: Halmazok (18 pont)

Az alábbi algoritmusok az A, B, C – rendre N, M, ill. P elemű – halmazok elemeiből állítják elő a D halmazt. A halmazokat tároló tömbökben az elemek növekvő sorrendben vannak, és A(N)=B(M)=C(P).

```
Első:
  I:=1; J:=1; K:=1; L:=0
  Ciklus amíg I≤N és J≤M és K≤P
     Elágazás
        A(I)<B(J) esetén I:=I+1
        B(J)<C(K) esetén J:=J+1
        C(K)<A(I) esetén K:=K+1
           egyéb esetben L:=L+1; D(L):=A(I)
                         I:=I+1; J:+J+1; K:=K+1
     Elágazás vége
  Ciklus vége
Eljárás vége.
```

```
Második:
  I:=1; J:=1; K:=1; L:=0
  Ciklus amíg I≤N és J≤M és K≤P
     Elágazás
       A(I)<min(B(J),C(K)) esetén L:=L+1; D(L):=A(I); I:=I+1
       B(J)<min(C(K),A(I)) esetén L:=L+1; D(L):=B(J); J:=J+1
       C(K)<min(A(I),B(J)) esetén L:=L+1; D(L):=C(K); K:=K+1
       egyéb esetben              (*)
            Elágazás
              A(I)<B(J) esetén I:=I+1; K:=K+1
              B(J)<C(K) esetén I:=I+1; J:=J+1
              C(K)<A(I) esetén J:=J+1; K:=K+1
                 egyéb esetben L:=L+1; D(L):=A(I)  (**)
                               I:=I+1; J:+J+1; K:=K+1
               Elágazás vége
     Elágazás vége
  Ciklus vége
Eljárás vége.
```

A. Milyen elemek kerülnek a D halmazba az egyes eljárások végrehajtásakor?

B. Mi az első eljárásban a ciklus minimális, illetve maximális lépésszáma rögzített N, M és P mellett?

C. A második eljárásban milyen feltétel teljesülése esetén hajtható végre a külső elágazás (*)-gal jelölt `egyéb esetben` ága, illetve a belső elágazás (**)-gal jelölt `egyéb esetben` ága?

5. feladat: Képfeldolgozás (12 pont)

Egy NxM-es térkép pontjainak magasságát 0 és 255 közötti fényességi kóddal ábrázoljuk az A mátrixban (a magasabban fekvő pontok fényesebbek). Az alábbi algoritmus a B, a C, illetve a D mátrixba e kép valamilyen transzformáltját rakja. Mely pontok lesznek az átlagosnál fényesebbek, melyek átlagosak és melyek sötétebbek a B, a C, illetve a D mátrixban tárolt képeken? (A mátrixok (1,1) koordinátájú pontja felel meg a képek bal felső sarkának.)

```
Képfeldolgozás:
  Ciklus I=2-től N-1-ig
     Ciklus J=2-től M-1-ig
       B(I,J):=128+(A(I,J)-A(I,J-1)) div 2
       C(I,J):=128+(2*A(I,J)-A(I,J-1)-A(I-1,J)) div 4
       D(I,J):=128+(4*A(I,J)-A(I,J+1)-A(I+1,J)-A(I,J-1)
                    -A(I-1,J)) div 8
     Ciklus vége
  Ciklus vége
Eljárás vége
```

6. feladat: Adatbázis-halmazok (20 pont)

Egymással összekapcsolt halmazokon értelmezünk műveleteket az alábbiak szerint:

Két halmaz elemei között a kapcsolatot egy vonallal jelöljük. A példában a **bal** oldali halmaz (Osztály) egy-egy eleméhez a **jobb** oldalinak (Diák) több eleme is tartozhat, a **jobb** oldali minden eleme a **bal** oldalinak pontosan egy eleméhez tartozik.

A halmazelemek jellemzőit a "/" után adjuk meg az ábrán. Jelölésük az algoritmusokban: Halmaz.jellemző (pl. Diák.Átlag az összes diák átlagát jelenti). Definiálhatók változók is, amelyek nem halmazhoz kapcsolódnak.

Új változót (és halmazjellemzőt) értékadással hozhatunk létre, formája:

```
Változó = FÜGGVÉNY Halmaznév (aritmetikai kifejezés) [logikai kifejezés]
```

A *változó* helyett *halmaznév.jellemző* is megadható. A logikai kifejezés (ha van) kiválogatja a halmazból a logikai kifejezést kielégítő elemeket, az aritmetikai kifejezés (ha van) ebből számít ki valamilyen értéket, és a függvényt (ha van) ezekre alkalmazzuk. **A kifejezésekben hivatkozni lehet az adott halmaz saját jellemzőire, a változókra, továbbá olyan halmazok jellemzőire, amelyek az adott halmaztól balra és vele összeköttetésben vannak**.

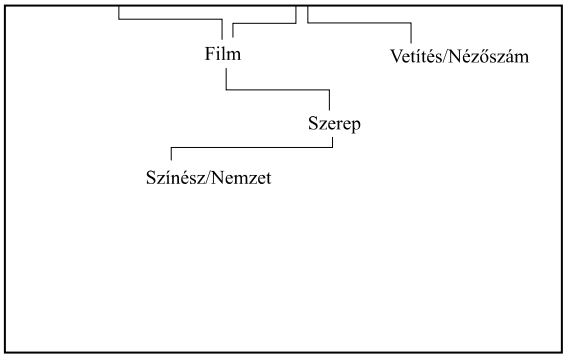
Az alkalmazható függvények:

ÖSSZEG, ÁTLAG, ELEMSZÁM, MAXIMUM, VAN, NINCS, MINDEN

Példa:

Osztály.fiú_átlag=ÁTLAG Diák (Átlag) [Fiú]	–az osztályok fiútanulói átlagának átlagai
Iskolai_átlag=ÁTLAG Diák (Átlag)	– iskolai átlag
Diák.eltérés=Diák (Átlag – Iskolai_átlag)	– az iskolai átlagtól eltérés diákonként
Osztály.mind_jobb=MINDEN diák [eltérés>0]	– azon osztályok, amelyekben minden diák átlaga jobb az iskolai átlagnál

Válaszolj az alábbi ábrához kapcsolódó kérdésekre!



A. Mit adnak meg a következő képletek?

Mozi.X=ÖSSZEG Vetítés (Nézőszám) [Rendező.Név="Rendes Ödön"]

Y=MAXIMUM Mozi (X)

Mozi.Z=Mozi (X=Y)

B. Mit adnak meg a következő képletek?

Műsor.X=ÖSSZEG Vetítés (Nézőszám)

Mozi.Y=ÁTLAG Műsor (X)

Film.Z=ELEMSZÁM Műsor [X>Mozi.Y]

C. Írj fel olyan képletet, amelyik filmenként megadja, hogy vetítették-e Budapesten, és egy másikat, amelyik megadja azokat a színészeket, akiknek minden filmjét vetítették Budapesten!

D. Írj fel olyan képletet, amelyik filmenként megadja, hogy van-e bennük japán szereplő, és egy másikat, amelyik megadja azokat a mozikat, ahol nem látható japán színész!

1997. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Határátkelő (24 pont)

Óriásországból Törpeországba egyetlen határátkelőhelyen, Manófalván keresztül lehet átjutni. A határon egyszerre egy autó léphet át, mivel mindig csak egy határőr van szolgálatban. Az útlevél- és vámvizsgálat minden autó esetén P percig tart.

Készíts programot, amely beolvassa P értékét, majd pedig az Óriásországból érkező autók határra érkezési idejét (óra, perc) növekvő sorrendben (legfeljebb 100 autó jön naponta). Az adatbevitelt -1-gyel zárjuk. Válaszul a program írja ki az egyes autók határra érkezési és Törpeországba való belépési idejét a példához hasonló módon!

Példa:

P = 30 perc

1. autó a határra érkezik: 10 óra 15 perckor, belép: 10.45-kor

2. autó a határra érkezik: 10 óra 55 perckor, belép: 11.25-kor

3. autó a határra érkezik: 11 óra 5 perckor, belép: 11.55-kor

4. autó a határra érkezik: 11 óra 10 perckor, belép: 12.25-kor

2. feladat: Mértékegységek (26 pont)

Angliában egészen más hosszmértékeket is használnak, mint nálunk:

- 1 league = 3 mile
- 1 mile = 8 furlong
- 1 furlong = 40 pole
- 1 pole = 5.5 yard
- 1 yard = 3 foot
- 1 foot = 12 inch
- 1 inch = 10 line
- 1 line = 2.54 mm

A magyarországi (metrikus) mértékegységek:

1 km = 1000 m 1 m = 10 dm 1 dm = 10 cm 1 cm = 10 mm

Készíts programot, amely

A. beolvas nyolc számot: egy hosszúságot az angol mértékrendszerben a hosszmértékek csökkenő sorrendjében, majd kiírja metrikus mértékegységekre átszámítva;

B. beolvas öt számot: egy hosszúságot a metrikus mértékrendszerben a hosszmértékek csökkenő sorrendjében, majd kiírja angol mértékegységekre átszámítva.

Példa:

Bemenet: 0 2 0 0 2 1 0 0

Eredmény: 3 km 220 m 8 dm 2 cm 2 mm

Bemenet: 3 220 8 2 2

Eredmény: 0 league 2 mile 0 furlong 0 pole 2 yard 1 foot 0 inch 0 line

(A mm, ill. a line lehet valós szám, a többi mértékegység pedig mindig egész.)

3. feladat: Karácsonyfa (25 pont)

A kukutyini általános iskolában karácsonyfát állítanak, melynek ágaira N db 1 kg-os óriás-szaloncukrot akasztanak. Minden szaloncukorról megadjuk, hogy a fa északi, keleti, déli vagy nyugati oldalára kerül-e. Ha a fa feldíszítése után valamelyik oldalon (észak, kelet, ...) legalább 25 %-kal több szaloncukor van, mint a szemben levő oldalon, akkor a fa abba az irányba eldől. Ha egyszerre két főégtájba is dőlnie kellene (pl. északra és keletre), akkor a közbülső irányba dől (pl. északkeletre). Készíts programot, amely beolvassa N (≤100) értékét, és az N db szaloncukor irányának betűjelét (E,K,D,N), majd kiírja, hogy eldől-e a fa, s ha igen, akkor merre.

Példák:

Bemenet: 6 E E D K N K Kimenet: A fa északkeletre dől

Bemenet: 6 E D E K N D Kimenet: A fa nem dől el

Kilencedik-tizedik osztályosok

1. feladat: Üvegválogatás (15 pont)

Egyforma rekeszekben fehér és színes üvegpalackok vannak. Minden rekesz tele van. Szét kell válogatni az üvegeket úgy, hogy legfeljebb egy rekeszben legyen fehér és színes üveg is, a többiben

vagy csak fehér, vagy csak színes. Egy lépésben egy fehér üveget cserélünk fel egy másik rekeszben lévő színes üveggel.

Készíts programot, amely kiszámítja, hogy minimálisan hány cserére van szükség az üvegek szétválogatásához.

Az UVEGx.BE állomány első sorában az egy rekeszben lévő üvegek száma (2≤M≤100), a második sorában a rekeszek száma (1≤N≤1000) van. A harmadik sor N számot tartalmaz: az egyes rekeszekben lévő fehér üvegek számát.

Az UVEGx.KI állományba és a képernyőre a szétválogatáshoz szükséges cserék minimális számát kell kiírni.

Példa:

UVEG1.BE	UVEG1.KI
5 6 2 0 3 5 2 4	3

2. feladat: Benzinkút (21 pont)

Egy benzinkútnál K töltőhelyen lehet tankolni. A kúthoz összesen N autó érkezik, melyek különböző ideig foglalják el a töltőhelyeket. Az érkező autók egy szabad töltőhelyhez állnak. Ha minden hely foglalt, akkor várakoznak, majd érkezési sorrendben állnak a szabaddá vált kutakhoz.

A BENZINx.BE állomány első sorában N (≤100) és K (≤10) értéke van, szóközzel elválasztva. A következő N sor az egyes autók érkezési (óra, perc), illetve tankolási (perc) idejét tartalmazza.

Készíts programot, amely a BENZINx.KI állományba és a képernyőre írja ki egy-egy sorba, hogy az egyes autók – érkezési idejük sorrendjében – mikor hagyják el a benzinkutat (óra, perc)!

Példa:

BENZIN1.BE	BENZIN1.KI
3 2 6 30 35 6 35 10 6 40 10	7 5 6 45 6 55

3. feladat: Vonatok (21 pont)

A Kukutyin-Piripócs útvonalon naponta egy-egy vonat közlekedik, az egyik Kukutyinból megy Piripócsra, a másik pedig Piripócsról Kukutyinba. Minden állomáson van kitérő, de az állomások közötti szakaszok csak egyetlen vágányból állnak. Egy szakaszon egyszerre csak egy vonat haladhat. Ha egy szakasz foglalt, a vonatnak az állomáson kell megvárnia a szembe jövő vonatot. Azonos továbbindulási idő esetén a Kukutyinból jövő vonatnak van elsőbbsége, a Piripócs felől érkezővel szemben.

A VONATx.BE állomány első sora a kukutyini indulást tartalmazza (óra és perc, egyetlen szóközzel elválasztva), a második pedig a piripócsit. A harmadik sorban a két végállomás közötti állomások száma (N≤100) található, a következő N+1 sorban pedig az I–1. és az I. állomás közötti távolság megtételéhez szükséges idő és az I. állomáson várakozással eltöltendő idő percben, szóközzel elválasztva.

Készíts programot, amely a VONATx.KI állományba és a képernyőre kiírja soronként az alábbiakat:

A. Annak az állomásnak a sorszámát, ahol a két vonat találkozik. (Kukutyin a 0., Piripócs az N+1. állomás.)

B. Annak a végállomásnak a nevét, amelyikből jövő vonatnak várakoznia kell a másikra. A sor legyen üres, ha nem kell várnia egyiknek sem

C. A várakozási időt percben.

D. A Kukutyinból jövő vonat indulási idejét az egyes állomásokról, valamint a végállomásra érkezés idejét (egy sorba, szóközökkel elválasztva).

E. A Piripócsról jövő vonat indulási idejét az egyes állomásokról, valamint a végállomásra érkezés idejét (egy sorba, szóközökkel elválasztva).

Példa:

VONAT1.BE

```
11 20
11 30
2
10 5
10 5
8
```

VONAT1.KI

```
2
Piripócs
2
11 20 11 35 11 50 11 58
11 30 11 45 12 0 12 10
```

4. feladat: Foltkód (18 pont)

Egy MxN pontból álló fekete-fehér képen egyetlen fekete folt lehet, ennek belsejében nincs fehér pont, s a folt a kép szélét nem éri el. A fekete pontok helyén 'X', a fehérekén '.' áll.

A foltot ún. lánckóddal írjuk le, ami kezdőpont koordinátáiból és a folt körüljárása során kapott ún. iránykód-sorozatból áll. A kezdőpont az a fekete pont, amelyet a kép bal felső sarkából kiindulva, soronként balról jobbra haladva elsőnek találunk meg. A kezdőpontból elindulva az óramutató járásával egyező irányban haladunk a folt peremén, amíg a lehető legrövidebb útvonalon vissza nem jutunk a kezdőpontba. Egy-egy pontból 8 irányba léphetünk, az irányokat az ábrán látható módon kódoljuk (ezek az ún. iránykódok).

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

A FOLTx.BE állomány első sora N (1≤N≤100) és M (1≤M≤100) értékét tartalmazza egy szóközzel elválasztva. A következő N sorban soronként M db karakter írja le a kép sorait: 'X' jelöli a fekete, '.' pedig a fehér pontokat.

Írj programot, amely a FOLTx.KI állományba és a képernyőre kiírja a folt lánckódját! Az első sorban kezdőpont sor- és oszlopindexe legyen, egy szóközzel elválasztva, a másodikban pedig a lánckód karakterei. Ha az adatok alapján nincs folt a képen, akkor a NINCS FOLT szöveget kell kiírni!

Példa:

FOLT0.BE

```
4 5
.....
..XX.
.XX..
.....
```

FOLT0.KI

```
2 3
7438
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Sereg (16 pont)

Burkusországban különös elven működik a hadsereg. Minden katonának 10 arany a zsoldja évente, plusz még annyi, ahány (nem csak közvetlen) beosztottja van. A sereg zsoldját a főparancsnok veszi át a királytól. A saját részét megtartja, a többit egy nap múlva adja tovább közvetlen alárendeltjeinek.

Ők is hasonlóan cselekszenek: a saját részüket megtartják, a többit újabb egy nap múlva adják tovább közvetlen alárendeltjeiknek s.í.t.

Az első év elején a sereg mindössze a leendő főparancsnokból áll.

A SEREGx.BE állomány 1. sorában a belépők száma (N≤100), a vizsgált időszak hossza (H) években, valamint a főparancsnok neve van, szóközzel elválasztva. A következő 3*N sorban a belépés sorrendjében a belépési év sorszámát (1-től kezdve), a belépő és közvetlen parancsnoka nevét adjuk meg.

Készíts programot, amely meghatározza, hogy H éven keresztül évente

A. mennyi zsoldot fizet a király a seregnek,

B. hány olyan katona van, akinek nincsen alárendeltje,

C. kinek van a legtöbb közvetlen alárendeltje (ha több ilyen van, akkor csak egyet kell megadni),

D. hányadik napon ér véget a zsoldosztás.

A SEREGx.KI állományba és a képernyőre a fenti 4 kérdésre adott válaszokat szóközzel elválasztva, H sorba kell kiírni; az i. sorba az i. évre vonatkozó választ.

Példa:

SEREG1.BE

```
3 2 Tábornok
1
Ezredes
Tábornok
1
Százados
Ezredes
2
Kapitány
Ezredes
```

SEREG1.KI

```
33 1 Tábornok 3
45 2 Ezredes 3
```

2. feladat: Járda (15 pont)

Egy gyalogjárda két sorban négyzet alakú betonlapokkal van fedve. A járdalapokat mindkét sorban balról jobbra 1-től N-ig számozzuk. A törötteket ki kell cserélni. Mivel csak 2x1 méretű új lapok állnak rendelkezésre, szomszédos lapokat kell kicserélnünk. Sajnos, felszedés közben a hibátlan lapok is eltörnek, így tovább nem használhatók.

	X	X	X					X	X
X			X	X	X		X	X	

(A törött lapokat X-szel jelöltük.)

Készíts programot, amely kiszámítja, hogy legalább hány új 2x1-es lap kell a javításhoz.

A JARDAx.BE állomány első sorában a járda lapokban mért hossza (2≤N≤1000) van. A következő két-két sor a törött lapok darabszámát és sorszámát adja meg:

a járda felső sorában:	a törött lapok száma az egyes törött lapok sorszáma szóközzel elválasztva
a járda alsó sorában:	a törött lapok száma az egyes törött lapok sorszáma szóközzel elválasztva

A JARDAx.KI állományba és a képernyőre a javításhoz minimálisan szükséges 2x1-es lapok számát kell írni.

3. feladat: Turista (19 pont)

Egy turista a turistaházból a lehető legrövidebb idő alatt szeretne eljutni a forráshoz. Útja hegyeken, völgyeken, szakadékokon vezet át. Minden út észak-déli vagy kelet-nyugati irányú. Előfordulhat, hogy a nagy szintkülönbség miatt kerülőt kell tennie. Segíts neki az útvonal megtervezésében!

Az úthálózat alkotta rácsot, ahol a rácspontok az útelágazások, egy NxM-es tömbbel ábrázoljuk; az egyes cellák értéke a domborzat adott rácspontban mért magassága. A turistaház a bal felső sarokban (az (1,1) koordinátájú pontban), a forrás a jobb alsó sarokban (az (N,M) koordinátájú pontban) található.

Egy útszakasz megtétele sík terepen egy percig tart. Minden méter szintkülönbség leküzdése további egy perccel növeli meg ezt az időt. Ha két pont szintkülönbsége nagyobb az előre megadott K értéknél, a turistának kerülőt kell tennie.

Írj programot, amely meghatározza a túra megtételéhez szükséges legrövidebb időt és az útvonal szakaszainak számát!

A HEGYx.BE állomány első sorában N, M és K értéke (1≤N≤100, 1≤M≤100) található, egy-egy szóközzel elválasztva. A következő N sor mindegyike M rácspont magasságát tartalmazza, szóközzel elválasztva.

A HEGYx.KI állományba és a képernyőre a túra megtételéhez szükséges legrövidebb időt és az útszakaszok számát kell írni! Ha nem lehet eljutni a célba, a program írja ki a NEM ÉR CÉLBA szöveget!

Példa: (vastagon szedve a túra útvonala)

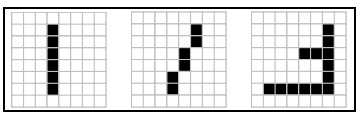
HEGY1.BE

```
5 5 100
100 110 150 200 250
200 120 130 200 100
200 200 140 150 200
200 200 150 190 190
200 200 160 170 180
```

HEGY1.KI

```
88 8
```

4. feladat: F.I.L.E (16 pont)



Az ábécé betűit 8x8-as pontmátrixban ábrázoljuk, '.'-tal kódoljuk a világos, és 'X'-szel a sötét pontokat. A vonalak belsejében levő sötét pontoknak legfeljebb 2 sötét szomszédja lehet, a vonalak találkozásánál levőknek 3, a vonalak végén levőknek pedig 1. A betűk tetszőleges állásúak lehetnek (pl. normál, ferde, oldalra döntött, fejre állított), és a felismerhetőség határán belül a méretük is különbözhet (pl. az I betű állhat 8 pontból, de akár 4-ből is).

Az I betű egyenes és döntött, az F fordított állásban látható az ábrán.

A BETUKx.BE állomány első sorában a betűk száma (N), a következő N*8 sorban pedig az egyes betűk pontmátrixa, soronként 8 karakter ('X' vagy '.') van. Az állományban csak a következő nagybetűk fordulnak elő:

```
BDEFHILOY.
```

Készíts programot, amely a bemeneti állományból - a vonalak száma és találkozásaik jellege alapján - felismeri az F, I, L, E betűket! A BETUKx.KI állományba és a képernyőre egyetlen sorba (a bemenő állomány sorrendjében) ki kell írni a felismert betűket. A fel nem ismert betűk helyére a – karakter írandó.

Példa:

```
BETUK1.BE          BETUK1.KI
2                  I-
........           mert az Y-t a programnak nem kell felismernie
........
........
........
.XXXXX..
........
........
........

........
.X.....X
..X...X.
...X.X..
....X...
....X...
....X...
........
```

1997. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Állatkert (25 pont)

A fokföldi állatkertben a ragadozó állatokat (maximum 50) szomszédos ketrecekbe rakták, minden ketrecbe egyet. Arra is ügyeltek, hogy az azonos fajú állatok egymás mellett legyenek. Készíts programot, amely a ketrecek sorrendjében beolvassa a bennük levő állatfajok nevét (soronként egyet, az utolsó után csak az ENTER-t kelljen még egyszer lenyomni), majd megadja, hogy hányféle állat, és melyikből mennyi van az állatkertben, valamint azt, hogy melyik állatból van a legtöbb!

2. feladat: Bank (20 pont)

Megtakarított pénzünket N éven keresztül egy bankba rakjuk az alábbi módon:

1. N éven keresztül minden hónap elején beteszünk A forintot, és egy évre X százalékos kamattal (X>0, valós szám) lekötjük.

2. A lekötés lejártakor, a következő hónap elején kapjuk meg az éves kamatot, ekkor a bent levő pénzt a kamattal együtt egy évre újra lekötjük.

Készíts programot, amely beolvassa N, A és X értékét, majd N éven keresztül minden hónap elején kiírja, hogy mennyi pénzünk van a bankban!

Példa: (N=3, A=1000, X=10 esetén)

1.év: 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000 11000 12000

2.év: 13100 14200 15300 16400 17500 18600 19700 20800 21900 23000 24100 25200

3.év: 26410 27620 28830 30040 31250 32460 33670 34880 36090 37300 38510 39720

3. feladat: Balaton (30 pont)

Tavaly nyáron minden délben megmértük a Balaton vizének hőmérsékletét. Készíts programot, amely beolvassa a nyári napok számát (maximum 100), az egyes napokon mért vízhőmérsékletet, majd megadja:

A. azoknak a napoknak a számát, amikor a Balaton vize legalább 23 fokos volt;

B. egy olyan (tetszés szerinti) 7 napos időszakot, amikor a víz folyamatosan legalább 23 fokos volt.

C. azt a leghosszabb időszakot, amikor a víz folyamatosan legalább 23 fokos volt.

Kilencedik-tizedik osztályosok

Feladat: Dél-Afrikai törzsek (75 pont)

A jelenlegi Dél-Afrikai Köztársaság területén számos törzs élt, amikor a fehér gyarmatosítók megjelentek. Közülük egyesek békében éltek, mások időről időre ellenséges viszonyba keveredtek vagy éppen harcban álltak szomszédaikkal.

A TORZSEK.BE állomány első sorában a vizsgált időszak kezdő és befejező éve, valamint a vizsgált törzsek száma (N≤20) található, a következő N sorában pedig az egyes törzsek neve (mivel nincs mindegyiknek magyar megfelelője, a példákban az angol nevüket használjuk).

A TORZSx.BE állományban tároljuk – kezdőévük növekvő sorrendjében – azt, hogy az egyes törzsek között mettől meddig tartott az ellenségeskedés. Minden egyes ellenséges viszonyt 4 adattal írunk le: megadjuk a két törzs nevét, továbbá az ellenségeskedés kezdő-, illetve befejező évét (maximum 100 az ellenségeskedések száma).

Készíts programot, amely a TORZSx.KI állományba és a képernyőre kiírja egy-egy sorba az alábbiakat:

A. A békés törzsek nevét, amelyek a vizsgált időszakban senkivel sem voltak ellenséges viszonyban.

B. Annak a két törzsnek a nevét, amelyek a leghosszabb ideig voltak egymással folyamatosan ellenséges viszonyban.

C. A leghosszabb békés időszakot, amikor mindenki mindenkivel békében élt.

D. Egy olyan évet, amikor a legtöbb törzs állt ellenséges viszonyban valamelyik másikkal.

E. Annak a törzsnek a nevét, amelyik egy időben a lehető legtöbb törzzsel állt ellenséges viszonyban.

F. Törzseknek azt a legnagyobb halmazát, amelyben bármely két törzs volt ellenséges viszonyban a vizsgált időszakban.

G. Törzseknek azt a legnagyobb halmazát, amelyben a vizsgált időszakban mindenki volt mindenkinek az ellensége, vagy az ellenségének az ellensége, vagy annak az ellensége ...

Tizenegyedik-tizenharmadik osztályosok

Feladat: Bandák (75 pont)

Johannesburg rendőrsége az összes bűnözőt ismeri a városban. Tudják, hogy két bűnöző akkor tartozik ugyanabba a bandába, ha legalább egyszer közösen követtek el bűncselekményt (a bandák tagjai közül csak a közös bűncselekményt elkövetők ismerik egymást).

A bűnözőket a nevük helyett a sorszámukkal azonosítják (1-től kezdve, folyamatos számozással). A rendőrség a BANDAx.BE szöveges állományban tárolja az egy bandába tartozó bűnözők sorszámát. Az állomány első sorában a bűnözők száma (maximum 50), minden további sorában (maximum 100) pedig két bűnöző sorszáma van (egyetlen szóközzel elválasztva), akik közösen követtek el bűncselekményt. Lehetnek olyan magányos bűnözők is, akik egyetlen bandának sem tagjai (ahol bandákkal kapcsolatos eredményt kérünk, ott őket nem szabad figyelembe venni).

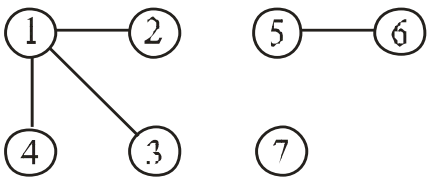
Példa:

BANDA0.BE

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

BANDA0.KI

```
7
2
1 5
4
1 5
0
2
1
```



Készíts programot, amely a BANDAx.KI állományba és a képernyőre kiírja egy-egy sorba az alábbiakat. (Ha valamelyik megoldás több számból áll, akkor közéjük egy-egy szóközt kell írni. Egy üres sort akkor is írj ki az állományba és a képernyőre, ha az adott részfeladattal nem foglalkoztál!)

A. A magányos bűnözők sorszámát. (A fenti példa szerint egyedül a 7. magányos.)

B. A bandák számát. (A példában 2, hiszen az egyikbe tartozik az 1., 2., 3. és 4., a másikba pedig az 5. és 6.; a magányos bűnözők nem alkotnak bandát.)

C. Az egyes bandák legtöbb kapcsolattal rendelkező tagját. (A példában 1., illetve 5. vagy 6.)

D. A legnagyobb banda tagjainak számát. (A fenti példa esetén 4, mivel az 1., 2., 3. és 4. bűnöző van egy bandában.)

E. Az egyes bandák kulcsembereit, azaz azokat, akiket börtönbe zárva a bandák a lehető legtöbb független részre esnek szét. (A példában 1., illetve 5. vagy 6.)

F. A biztonságosan szervezett bandák számát, azaz az olyanokét, amelyekből a rendőrség nem tud úgy letartóztatni egyetlen bűnözőt, hogy a banda több részre essen szét. (A példában 0, hiszen az 1. bűnöző letartóztatásával az 1. banda magányos bűnözőkre esik szét, a kéttagú bandák egyik tagjának letartóztatásával pedig a másik tag ugyancsak magányos bűnözővé válik.)

G. A kockázatosan szervezett bandák számát, azaz az olyanokét, amelyekben csak a főnök ismer mindenkit, és ezért az ő letartóztatása esetén a banda magányos bűnözőkre esik szét. (A példában 2, hiszen az 1. bűnöző letartóztatásával a 2., 3. és 4. magányos bűnözővé válik, a kéttagú bandák pedig mindig ilyenek.)

H. A totálisan szervezett bandák számát, azaz az olyanokét, amelyekben mindenki ismer mindenkit. (A példában 1, hiszen a kéttagú bandák mindig ilyenek.)

A verseny végeredménye:

I. kategória

1. Felföldi Zsolt	Fazekas Mihály Gimnázium, Budapest
1. Rokob András	Szilágyi Dezső Általános Iskola, Miskolc
2. Ritter Ádám	Fazekas Mihály Gimnázium, Budapest
3. Soltész Péter	Hungária körúti Általános Iskola, Budapest
4. Csirmaz Előd	Fazekas Mihály Gimnázium, Budapest
Zavarkó Gábor	Földes Ferenc Gimnázium, Miskolc
Szeti Balázs	Teleki Blanka Gimnázium, Székesfehérvár
Pápai Gábor	Türr István Gimnázium, Pápa
8. Balogh János	Kinizsi lakótelepi Általános Iskola, Kaposvár
9. Kormos Gergő	Bolyai János Gyakorló Iskola és Gimnázium, Szombathely

10. Pallos Péter	Fazekas Mihály Gimnázium, Budapest
II. kategória	
1. Felföldi Zsolt	Fazekas Mihály Gimnázium, Budapest
2. Rácz Balázs	Veres Péter Gimnázium, Budapest
3. Németh András	Fazekas Mihály Gimnázium, Budapest
4. Csillag Kristóf	Karacs Ferenc Gimnázium, Püspökladány
Gera Zoltán	Neumann János Szakközépiskola, Budapest
6. Merksz Andor	Bencés Gimnázium, Pannonhalma
Helmecz Vajk	Veres Pálné Gimnázium, Budapest
Zupán Kristóf	Piarista Gimnázium, Budapest
9. Fazekas Dániel	Révai Miklós Gimnázium, Győr
10. Pintér Lóránt	Táncsics Mihály Gimnázium, Kaposvár
Németh András	Árpád Gimnázium, Budapest
III. kategória	
1. Ujhelyi Gábor	Földes Ferenc Gimnázium, Miskolc
Lövey László	Neumann János Szakközépiskola, Eger
Várkonyi Dániel	Teleki Blanka Gimnázium, Székesfehérvár
4. Végh Dávid	Fazekas Mihály Gimnázium, Budapest
5. Várnagy Zoltán	Bólyai János Gimnázium, Salgótarján
Marhefka István	Avasi Gimnázium, Miskolc
Tóth László	Földes Ferenc Gimnázium, Miskolc
Papp Márton	Radnóti Miklós Gimnázium, Dunakeszi
Bedő Sándor	Zrínyi Miklós Gimnázium, Zalaegerszeg
10. Jenei Attila	Apáczai Csere János Gimnázium, Pécs

1998. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Karikák (25 pont)

Az alábbi Logo programrészletek köröket rajzolnak különböző elrendezésben. A `kör 10` eljárás-hívás 10 egység sugarú kört rajzol, az aktuális pontot véve középpontnak. A rajzolás elején a teknőc északi irányba néz, a toll a levegőben van, a `kör` eljárás a körvonal rajzolása előtt leteszi a tollat, a végén pedig újra felemeli. Rajzolás után a teknőc visszaáll eredeti pozíciójába és irányába. Az `előre t` utasítás hatására a teknőc az aktuális irányba előre lép `t` egységgel, a `jobbra sz` hatására pedig jobbra fordul `sz` fokkal. Az `ismétlés db [utasítások]` hatására a záró-jelbe tett utasításokat db-szer megismétli.

Példa:

`kör 10 előre 10 kör 20` eredménye:

Melyik programrészlet mit rajzol?

```
A. ismétlés 4 [kör 10 előre 20 jobbra 90]
B. ismétlés 3 [kör 10 előre 20 jobbra 120]
C. ismétlés 4 [kör 10 előre 10 jobbra 90]
D. ismétlés 4 [kör 10 előre 15 jobbra 90]
E. jobbra 90 ismétlés 2 [kör 10 előre 15] kör 10
   jobbra 120 előre 15 kör 10 jobbra 60 előre 15
   kör 10 előre 15
```

2. feladat: Betűkirakó (25 pont)

Adott egy karaktersorozat és néhány 2+1 db karakterből álló szabály. Az elsőtől kezdve egyesével megnézzük a karaktersorozat elemeit, s ha valamelyik elem bal szomszédja valamelyik szabály első karaktere, jobb szomszédja pedig a második karaktere, akkor az elemet a szabály 3. karakterével helyettesítjük. (Az első elem bal szomszédja az utolsó elem, az utolsó elem jobb szomszédja az első elem.) A karaktersorozat utolsó eleme után a következő körben újra az első karaktertől folytatjuk.

Példa:

A karaktersorozat: 0101 A szabályok: (00,0), (01,1)

Ha 2 kört kell megtenni, a karaktersorozat így változik:

induló helyzet:	0101
1. kör:	**0**101,0**0**01,00**1**1,001**1**
2. kör:	**0**011,0**1**11,01**1**1,011**1**

A. Mi lesz az `AABBAB` sorozatból 2 kör megtétele után az `(AB,B) (BB,A) (AA,A)` szabályok alkalmazásával? Írd le lépésenként a kapott karaktersorozatot!

B. Írj olyan szabályokat, amelyekkel az `1000` karaktersorozatból `1997` lesz egy körben!

3. feladat: Fa (25 pont)

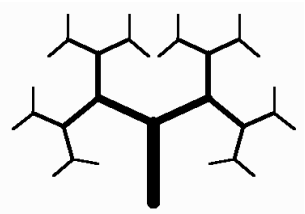
A Logo-ban a `tollvastagság! :x` utasítás a vonalvastagságot állítja be. Az elágazás `ha feltétel [az akkor ág utasításai]`, az eljárás-definíció

```
tanuld eljárásnév :paraméter
  utasítások
vége
```

alakú. Az alábbi Logo program egy fát rajzol a képernyőre.

```
tanuld fa :db :h :v
  tollvastagság! :v előre :h
  ha :db>1 [balra 60 fa :db-1 :h*3/4 :v*2/3
            jobbra 120 fa :db-1 :h*3/4 :v*2/3
            balra 60]
  tollvastagság! :v hátra :h
vége
```

A `fa` eljárást `:db :h :v` paraméterekkel hívjuk meg (`:db`>0).



A. Add meg a lehető legegyszerűbb képlettel, hogy a kezdő hívással együtt hányszor hívja meg saját magát az eljárás!

B. Hány levele (olyan végződése, ahonnan nem rajzolunk további ágat) lesz a fának ?

C. Mekkora felületet fednek le a fa vonalai (nem kell azzal törődnöd, hogy egyes ágakat egymásra rajzol-e az eljárás)?

4. feladat: Mit csinál? (25 pont)

Az alábbi algoritmus az `N` elemű `A` vektor és a `D` változó alapján határozza meg `L` és `S` értékét.

```
Valami:
  I:=1; K:=0
  Ciklus amíg I≤N és K<D
     Ha A(I) páros akkor K:=K+1
     I:=I+1
  Ciklus vége
  L:=K≥D
  Ha L akkor S:=I-1
Eljárás vége.
```

A. Mi lesz `L` értéke a futás végén?

B. Mi a szerepe a `K` változónak?

C. Mi a kezdeti feltétele annak, hogy az `S` változó kapjon értéket, s mi lesz ez az érték?

Kilencedik-tizedik osztályosok

1. feladat: Karikák (15 pont)

Az alábbi Logo programok köröket rajzolnak különböző elrendezésben. A `kör 10` eljáráshívás 10 egység sugarú kört rajzol, az aktuális pontot véve középpontnak. A rajzolás elején a teknőc északi irányba néz, a toll a levegőben van, a `kör` eljárás a körvonal rajzolása előtt leteszi a tollat, a végén pedig újra felemeli. Rajzolás után a teknőc visszaáll eredeti pozíciójába és irányába. Az `előre t` utasítás hatására a teknőc az aktuális irányba előre lép `t` egységgel, a `jobbra sz` hatására pedig jobbra fordul `sz` fokkal. Az `ismétlés db [utasítások]` hatására a zárójelbe tett utasítások :db-szer megismétli. Az elágazás `ha feltétel [az akkor ág utasításai]`, az eljárás-definíció

```
tanuld eljárásnév :paraméter
  utasítások
vége
```

alakú.

Melyik program mit rajzol?

```
A. tanuld nta :db
     ha :db > 0
        [ismétlés 2
                  [ismétlés :db [kör 10 előre 20]
                   jobbra 90]
         nta :db - 1]
   vége
B. tanuld ntb :db
     ha :db > 0
        [ismétlés :db [kör 10 előre 20]
         jobbra 120 ntb :db - 1]
   vége
C. tanuld ntc :db
     ha :db > 0
        [ismétlés 3
                  [ismétlés :db - 1 [kör 10 előre 20]
                   kör 10 jobbra 60 előre 20]
         ntc :db-1]
   vége
```

2. feladat: Halmazvarázs (12 pont)

Az A és B halmazok egész számokat tartalmaznak. Használni fogjuk a következő utasításokat:

`nemüres(H)`	igaz, ha a halmazban van elem.
`üreshalmaz`	üres halmazt ad eredményül.
`eleme(e,H)`	igaz, ha az `e` elem benne van a `H` halmazban.
`egyeleme(H)`	a `H` halmaz egy tetszőleges eleme.
`berak(e,H)`	az `e` elemet berakja a `H` halmazba.
`kivesz(e,H)`	kiveszi az `e` elemet a `H` halmazból.

```
Valami:
  H:=üreshalmaz
  Ciklus amíg nemüres(A) vagy nemüres(B)
     Ha nemüres(A) akkor
        e:=egyeleme(A); kivesz(e,A)
        Ha eleme(e,B) akkor berak(e,H); kivesz(e,B)
     Elágazás vége
     Ha nemüres(B) akkor
        e:=egyeleme(B); kivesz(e,B)
        Ha eleme(e,A) akkor berak(e,H); kivesz(e,A)
     Elágazás vége
  Ciklus vége
Eljárás vége.
```

A. Mit csinál a fenti program? Mi lesz `H` értéke a futás végén?

B. Milyen hatása lenne, ha a ciklusfeltételben szereplő `vagy` művelet helyére `és` műveletet írnánk? Miért?

C. Milyen hatása lenne az aláhúzott utasítások elhagyásának? Miért?

D. Mivel lehetne a fentieken kívül egyszerűbbé és egyúttal gyorsabbá tenni az algoritmust? Fogalmazd meg saját szavaiddal!

3. feladat: Számolgatós (24 pont)

Az alábbi algoritmusokban a legalább N+1 elemű, 1-től indexelt X vektor egy tizedes tört törtrészének számjegyeit tartalmazza. Például 0.75 esetén X(1)=7, X(2)=5.

```
Első(I):
  Ha I=N+1 akkor Ha X(I)>5 akkor X(I-1):=X(I-1)+1
  Ha I<N+1 akkor Első(I+1)
                 Ha X(I)>9 akkor X(I):=0; X(I-1):=X(I-1)+1
Eljárás vége.

Második(I):
  Ha I=N+1 akkor X(I-1):=X(I-1)+1
  Ha I<N+1 akkor Második(I+1)
                 Ha X(I)>9 akkor X(I):=0; X(I-1):=X(I-1)+1
Eljárás vége.

Harmadik(I):
  Ha I=N+1 akkor X(I-1):=10-X(I-1)
  Ha I<N+1 akkor Harmadik(I+1); X(I-1):=9-X(I-1)
Eljárás vége.
```

A. Mit csinál a fenti három algoritmus, ha az I=1 értékkel hívjuk meg, s N értéke 10?

B. Tetszőleges N esetén bizonyos esetekben hibásan működnek. Milyen X vektorok esetén?

4. feladat: Cserebere (18 pont)

Egy automata a bemenetére érkező jelsorozatból bizonyos szabályok szerint képzett más jelsorozatot állít elő. Minden egyes szabályt egy-egy jelsorozat-pár ír le. A pár első tagja a helyettesítendő, a második tagja a helyettesítő jelsorozat. Egy lépésben az automata megvizsgálja, hogy van-e olyan szabály, amelynek első tagja illeszthető az átalakítandó jelsorozatra, s közülük az elsőt alkalmazza; ha a helyettesítendő jelsorozat több helyen is illeszkedik, a cserét balról az első helyen hajtja végre. Ezt a lépést addig ismétli, amíg van alkalmazható szabály (a hasonlítást újra az átalakítandó sorozat első elemétől kezdi).

Példa:

Bemenet:	ARARAT
Szabályok:	(ARA,BA), (BB,BO)
Lépések:	ARARAT → BARAT → BBAT → BOAT
Kimenet:	BOAT

Add meg az alábbi feladatok mindegyikéhez a lehető legkevesebb szabályból álló megoldást! (A betűk mellett az = és a + is része a jelsorozatoknak. A darabszám azt jelenti, hogy az adott jelek számát előre nem ismerjük, a megoldásnak az összes ilyen szerkezetű jelsorozatra működnie kell.)

A. Bemenet: `AAABB=` (n db `A`, k db `B`)

Kimenet: `=CCCCC` (n+k db `C`)

B. Bemenet: `CABCCBAAB=+` (n db `A`, n db `B` és n db `C`)

Kimenet: `AAA=BBB+CCC` (a betűk száma változatlan)

C. Bemenet: `AAAABBBBBABA` (n db `A` és `B` tetszőlegesen összekeverve, A-val kezdve)

Kimenet: `ABABABABABAB` (a betűk száma változatlan)

5. feladat: Keresés mátrixban (16 pont)

Adott egy N*M-es A mátrix: $\begin{pmatrix} A_{1,1} & \dots & A_{1,M} \\ \dots & \dots & \dots \\ A_{N,1} & \dots & A_{N,M} \end{pmatrix}$

A mátrix milyen elhelyezkedésű elemének indexeit adják meg kimenő paraméterként az alábbi algoritmusok? Rajzold le a keresési irányt! Mikor működnek hibásan?

```
AKeres(I,J):
  I:=1; J:=1
  Ciklus amíg A[I,J]≠0
     Ha J<M akkor J:=J+1 különben I:=I+1; J:=1
  Ciklus vége
Eljárás vége.
```

```
BKeres(I,J):
  I:=1; J:=M
  Ciklus amíg A[I,J]≠0
     Ha I<N akkor I:=I+1 különben J:=J-1; I:=1
  Ciklus vége
Eljárás vége.
```

```
CKeres(I,J):
  I:=N; J:=M
  Ciklus amíg A[I,J]≠0
     Ha J>1 akkor J:=J-1 különben I:=I-1; J:=M
  Ciklus vége
Eljárás vége.
```

6. feladat: Kincskereső (15 pont)

A Sötét Erdő rablói minden útelágazásnál és minden ösvény végén pénzt rejtettek el. Az erdőbe pontosan egy út vezet, az út minden elágazásnál két irányban folytatódik. Az alábbi programrészletben az A 2^{N-1}-1 elemű tömb, az S 2^N-1 elemű tömb és az N érték a függvények bemenő paraméterei.

```
f1(A,S,N):
  p:=1; z:=S[p]
  Ciklus i=1-től N-1-ig
     Ha A[p] akkor p:=2*p különben p:=2*p+1
     z:=z+S[p]
  Ciklus vége
  f1:=z
Függvény vége.
```

```
f2(S,N):
  p:=1; q:=S[p]
  Ciklus i=2-től 2^N-1-ig
     Ha S[i]>q akkor p=i; q:=S[i]
  Ciklus vége
  z:=0
  Ciklus amíg p>1
     z:=z+1; p:=p div 2
  Ciklus vége
  f2:=z
Függvény vége.
```

A. Mi van az A és az S tömbben?

B. Mit jelent az `N` állandó?

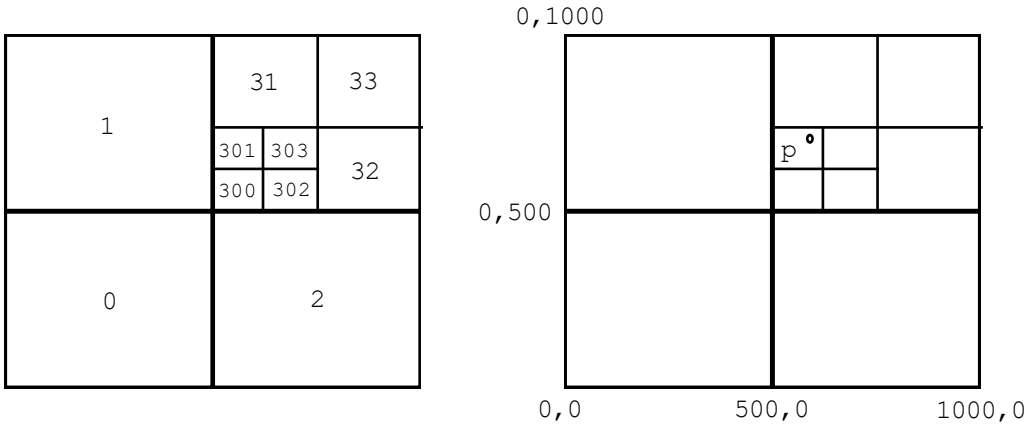
C. Mit csinálnak az egyes függvények?

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Síkbeli kód (14 pont)

Egy téglalap pontjait fix hosszúságú, négyes számrendszerbeli számokkal kódoljuk.

Első lépésben a teljes téglalapot – oldalainak felezésével – négy egyforma részre (síknegyedre) osztjuk. Minden síknegyedhez a bal oldali ábra szerinti kódot rendeljük. A síknegyedek további felosztását és kódolását hasonlóképpen folytatjuk, amíg el nem érünk egy előre megadott mélységet. Egy pont kódja a pontot magába foglaló síknegyedek kódjának sorozata. Egy határpontot a (leg)kisebb kódú síknegyedbe tartozónak tekintünk.



A jobb oldali ábrán az (x, y) = (610,720) koordinátájú pont 3 mélységű kódjának (301) meghatározása látható.

A. Határozd meg a [1-1000]×[1-1000]-es tartományba eső alábbi pontok 3 mélységű kódját:

1: (100,200) 2: (611,345) 3: (875,475) 4: (250,750)

B. Az alább felsorolt kódú pontok közül melyekről dönthető el egyértelműen, hogy a 600≤x<900, 100≤y<800 tartományba esnek?

1:001	2:003	3:011	4:021	5:023
6:102	7:112	8:123	9:130	10:131
11:200	12:203	13:211	14:221	15:232
16:233	17:302	18:303	19:312	20:320

2. feladat: Halmazok uniója (15 pont)

Az alábbi algoritmus három, tömbként ábrázolt halmaz (A,B,C) unióját állítja elő egy negyedikben (D). Az egyes halmazok elemszáma rendre AN, BN, CN, illetve DN.

```
Unió:
  D:=A; DN:=AN
  Ciklus I=1-től BN-ig
     J:=1
     Ciklus amíg J<AN és A(J)≠B(I)
        J:=J+1
     Ciklus vége
     Ha J≥AN akkor DN:=DN+1; D(DN):=B(I)
  Ciklus vége
```

```
  Ciklus I=1-től CN-ig
     J:=1
     Ciklus amíg J<AN és A(J)≠C(I)
        J:=J+1
     Ciklus vége
     Ha J≥AN akkor DN:=DN+1; D(DN):=C(I)
  Ciklus vége
Eljárás vége.
```

A. Milyen bemenő adatokra ad rossz eredményt az algoritmus?

B. Magyarázd el, hogyan lehet kijavítani az algoritmust?

3. feladat: Bizonytalan (10 pont)

A *BIZONYTALAN* programnyelv kétféle vezérlési szerkezetet ismer: az elágazást és a ciklust. Az elágazás egy igaz feltételű utasítást választ ki és hajt végre (több igaz feltétel közül a választás véletlenszerű).

IF feltétel$_1$ → utasítás$_1$ | feltétel$_2$ → utasítás$_2$ | ... | feltétel$_n$ → utasítás$_n$ FI

A ciklusban a ciklusmagot addig kell ismételni, amíg legalább egy feltétel teljesül. A ciklusmag egyszeri végrehajtása az egyik igaz feltételű utasítás végrehajtását jelenti (több igaz feltétel közül a választás véletlenszerű).

DO feltétel$_1$ → utasítás$_1$ | feltétel$_2$ → utasítás$_2$ | ... | feltétel$_n$ → utasítás$_n$ OD

Mit csinálnak az alábbi programrészletek?

A. `X`, `Y` és `Z` 1-nél nagyobb természetes számok.

```
A:=X; B:=Y; C:=Z
DO A>B → A:=A-B | B>C → B:=B-C | C>A → C:=C-A OD
```

B. Az `A` mátrix egyik oszlopában sem fordul elő ugyanaz az elem kétszer. X és Y tetszőleges, előre meghatározott értékek.

```
I:=1; J:=1
DO A[1,I]≠X → I:=I+1 | A[2,J]≠Y → J:=J+1 OD
VALASZ:=(I=J)
```

4. feladat: Cserebere (18 pont)

Egy automata a bemenetére érkező jelsorozatból bizonyos szabályok szerint képzett más jelsorozatot állít elő. Minden egyes szabályt egy-egy jelsorozat-pár ír le. A pár első tagja a helyettesítendő, a második tagja a helyettesítő jelsorozat. Egy lépésben az automata megvizsgálja, hogy van-e olyan szabály, amelynek első tagja illeszthető az átalakítandó jelsorozatra, s közülük az elsőt alkalmazza; ha a helyettesítendő jelsorozat több helyen is illeszkedik, a cserét balról az első helyen hajtja végre. Ezt a lépést addig ismétli, amíg van alkalmazható szabály (a hasonlítást újra az átalakítandó sorozat első elemétől kezdi).

Példa:

Bemenet:	ARARAT
Szabályok:	(ARA,BA), (BB,BO)
Lépések:	ARARAT → BARAT → BBAT → BOAT
Kimenet:	BOAT

Add meg az alábbi feladatok mindegyikéhez a lehető legkevesebb szabályból álló megoldást! (A betűk mellett az = és a + is része a jelsorozatoknak. A darabszám azt jelenti, hogy az adott jelek számát előre nem ismerjük, a megoldásnak az összes ilyen szerkezetű jelsorozatra működnie kell.)

A. Bemenet: `AAABB=` (n db A, k db B)

Kimenet: `=CCCCC` (n+k db C)

B. Bemenet: `CABCCBAAB=+` (n db A, n db B és n db C)

Kimenet: `AAA=BBB+CCC` (a betűk száma változatlan)

C. Bemenet: `AAAABBBBBABA` (n db A és B tetszőlegesen összekeverve, A-val kezdve)

Kimenet: `ABABABABABAB` (a betűk száma változatlan)

5. feladat: Speciális mátrixok (16 pont)

Egy speciális kitöltésű N*N-es mátrixot tömörítve például egy vektorral és egy függvénnyel ábrázolhatunk. A **mátrixban** az azonos betűk azonos értékű elemeket jelölnek, * jelöli azokat az elemeket, amelyek értéke közömbös, mert a felhasználásukra sohasem kerül sor. A mátrix bal felső elemének indexe (1,1). A **vektorban** az azonos betűvel jelölt azonos értékű elemek közül csak egyet tárolunk, a *-gal jelölt közömbös értékű elemeket pedig nem tároljuk. A vektor elemeit 1-től indexeljük. A **függvény** a mátrix egy indexpárját a vektor egy indexére képezi le.

Példa:

A mátrix: $\begin{pmatrix} a & a \\ b & b \end{pmatrix}$ A vektor: $(a \quad b)$ A függvény: `Index(I,J):=I`

Az alábbi négy feladat megoldására készíts egy-egy aritmetikai kifejezésből álló olyan függvényt, amely kiszámítja a mátrix `I`. sorának `J`. oszlopában levő elem vektorbeli indexét!

A. feladat

$$\begin{pmatrix} d & e & f & g \\ c & d & e & f \\ b & c & d & e \\ a & b & c & d \end{pmatrix} \qquad (a \quad b \quad c \quad d \quad e \quad f \quad g)$$

B. feladat

$$\begin{pmatrix} a & b & c & d \\ b & c & d & e \\ c & d & e & f \\ d & e & f & g \end{pmatrix}$$

C. feladat

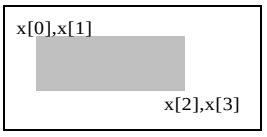
$$\begin{pmatrix} a & b & * & * \\ c & d & e & * \\ * & f & g & h \\ * & * & i & j \end{pmatrix} \qquad (a \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad i \quad j)$$

D. feladat

$$\begin{pmatrix} a & b & d & g \\ * & c & e & h \\ * & * & f & i \\ * & * & * & j \end{pmatrix}$$

6. feladat: Ablakok (27 pont)

Az `a` vektorban `n` db téglalap alakú ablak képernyő-koordinátáit tároljuk. `a`-ban takarási sorrendben vannak az ablakok: `a[1]`-ben a legalsó, `a[n]`-ben a legfelső. Egy téglalapot négyelemű vektorral adunk meg az ábrán látható módon. Az `x` téglalap „üres”, azaz nincs kirajzolandó pontja, ha `x[0]>x[2] vagy x[1]>x[3]`.



A következő programrésznek az a feladata, hogy a képernyőn az `x` téglalappal megadott területet, amely egy vagy több ablaknak lehet része, frissítse. (Egy pontot (pixelt) csak egyszer gyújt ki. Így villogásmentes és gyors lesz a rajzolás.) `Frissít(x,m)` kirajzolja az `a[m]` ablaknak az `x` téglalappal közös pontjait.

```
R(x,m):
  Ha m>0 akkor
     k:=1; Frissít(x,m)
     Ciklus i=0-tól 3-ig
       h:=x; k:=-k
       Ha k=1 akkor ertek:=max(h[(i+2) mod 4],a[m][i]+k)
            különben ertek:=min(h[(i+2) mod 4],a[m][i]+k)
       h[(i+2) mod 4]:=ertek
       R(h,m-1); k:=-k
       Ha k=1 akkor ertek:=max(x[i],a[m][i])
            különben ertek:=min(x[i],a[m][i])
       x[i]:=ertek
       k:=k*(1-(i mod 2)*2)
     Ciklus vége
  Elágazás vége
Eljárás vége.
```

A `k,i,h,ertek` az `R` lokális változói.

A. Az `m`. ablak vizsgálata közben, a ciklusmagba belépve, hogyan változik a `k` értéke? Az `a[m]` ablakhoz képest milyen területek frissítésére kerül sor `k` értékétől függően?

B. Az ábra segítségével rajzold le, hogyan változik az `x` paraméter által megadott terület az `m`. ablak vizsgálata közben, ha az `a[m]` teljes egészében része az `x`-nek! (Az ábrán az `m`. ablak az 5-össel jelölt mező, az `x` kezdetben pedig mind a kilenc mező.)

C. Mi történik, ha valamelyik lépésben az `m`. ablaknak és a frissítendő `x` területnek nincs közös része?

D. Mi lesz `x`-ben a ciklusból kilépve?

E. Hogyan lehetne az `m>0` feltételen változtatva gyorsítani az algoritmuson?

1998. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendező-pályaudvar (19 pont)

Egy kisvárosi rendező-pályaudvaron mindössze két vágányt használhatnak a továbbinduló szerelvények összeállításához. Egy-egy vágányon gyűjtik az azonos célállomásra küldendő vagonokat. Egy vágányon maximum M (≤50) vagon fér el. Ha egy vágány megtelt, vagy ha már mindkét vágányon állnak vagonok, s olyan vagon érkezik, amelyet egy harmadik helyre kell küldeni, akkor a két várakozó szerelvény közül a hosszabbat útnak indítják, s a helyére tolják az újonnan érkező vagont. Az utolsónak érkező vagon után a még várakozó szerelvényeket is elindítják, először (ha van) a hosszabbat.

Készíts programot, amely beolvassa a vagonok számát (N≤1000), a vágányokon elhelyezhető kocsik számát (M), majd az N vagon célállomását a pályaudvarra érkezésük sorrendjében. A program írja ki a szerelvények indulási sorrendjében, hogy melyik állomásra hány kocsiból álló szerelvényt kell indítani.

2. feladat: Állatkertek (27 pont)

A kukutyini állatkert N, a rátóti pedig M fajta (N,M≤100) állatot tart. Tudjuk, hogy melyik állatkertben melyik fajtából hány példány van. A két állatkert olyan állatokat cserélhet egymással, amelyekből

mindkettőnek legalább K példánya van, illetve olyan állatokat ajándékozhat a másiknak, amelyekből legalább L példánya van, a másiknak pedig egyetlen példánya sincs.

Készíts programot, amely beolvassa a kukutyini, majd a rátóti állatkert adatait, végül pedig K és L értékét! Mind a két esetben be kell olvasni az állatfajták számát, majd az egyes állatfajták nevét és példányszámát. A program írja ki, hogy mely állatfajtákból cserélhet a két állatkert, illetve mely állatfajtákból ajándékozhat a kukutyini a rátótinak, s melyekből a rátóti a kukutyininak!

Példa:

```
A kukutyini fajták száma?   A rátóti fajták száma? 5
4
                            1. állat? kecske
1. állat? kecske            példányszáma? 8
példányszáma? 12
                            2. állat? pulyka
2. állat? nyúl              példányszáma? 16
példányszáma? 41
                            3. állat? szamár
3. állat? ló                példányszáma? 1
példányszáma?8
                            4. állat? ló
4. állat? liba              példányszáma? 4
példányszáma? 76
                            5. állat? liba
                            példányszáma? 60
Mekkora létszám fölött lehet cserélni? 6
Mekkora létszám fölött lehet ajándékozni? 12
Csere:            Kukutyin ajándékoz:   Rátót ajándékoz:
  kecske            nyúl                  pulyka
  liba
```

3. feladat: Titkosírás (29 pont)

Az egyik legrégibb titkosírás szerint a titkosítandó szöveg minden betűjét az ábécé következő betűjével helyettesítjük. (Az utolsó betűt az első követi, azaz ha például az angol ábécét használjuk, a *z*-ből *a* lesz.) Az egyéb karaktereket változatlanul kell hagyni.

Készíts programot, amely beolvas egy ***kisbetűkből*** és egyéb karakterekből álló szöveget, ennek minden kisbetűjét helyettesíti a ***kiterjesztett magyar*** ábécé következő betűjével, majd kiírja a „titkosított" szöveget a képernyőre! A hosszú mássalhangzókat két betűnek (pl. *ssz* helyett *s* és *sz*), a (balról jobbra haladva) többjegyű betűnek is tekinthető betűsorozatokat többjegyűeknek vegye (pl. a *malacság* szóban a *cs*-t tekintse többjegyű betűnek)!

A kiterjesztett magyar ábécé: *a á b c cs d dz dzs e é f g gy h i í j k l ly m n ny o ó ö ő p q r s sz t ty u ú ü ű v w x y z zs*

Példa:

Bemenet:	Kimenet:
A bodza?	A códzsá?
Igazság	Igyáagy
liszteszsák	lyíttyétszbl

Kilencedik-tizedik osztályosok

1. feladat: Vetésterület (16 pont)

Egy téglalap alakú birtokon többféle növényt vetettek. A terület minden négyzetméteréről tudjuk, hogy ott milyen növény található. Növénytáblának azt a lehető legnagyobb, téglalap alakú területet nevezzük, amelyen belül csupa azonos növény nő.

A BIRTOKx.BE állomány első sora tartalmazza a birtokot leíró téglalap sorainak (1≤N≤50), illetve oszlopainak számát (1≤M≤50) egyetlen szóközzel elválasztva. A további N*M sorban a vetett növények neve található, sorfolytonosan.

Készíts programot, amely kiírja a képernyőre és a BIRTOKx.KI állományba a birtokon levő növénytáblák számát!

Példa:

BIRTOK0.BE:

```
4 5
Búza
Búza
Búza
Burgonya
Burgonya
Búza
Búza
Búza
Burgonya
Burgonya
Búza
Búza
Kukorica
Kukorica
Kukorica
Búza
Búza
Kukorica
Kukorica
Kukorica
```

A bemenet mátrix alakban:

Búza	Búza	Búza	Burgonya	Burgonya
Búza	Búza	Búza	Burgonya	Burgonya
Búza	Búza	Kukorica	Kukorica	Kukorica
Búza	Búza	Kukorica	Kukorica	Kukorica

A növénytáblák a birtokon:

Búza	Búza	Búza	Burgonya	Burgonya
Búza	Búza	Búza	Burgonya	Burgonya
Búza	Búza	Kukorica	Kukorica	Kukorica
Búza	Búza	Kukorica	Kukorica	Kukorica

BIRTOK0.KI:

```
4
```

2. feladat: Határőrség (25 pont)

Bergengócia lakosait nevük egyértelműen azonosítja, közülük legfeljebb 500-nak van útlevele. Az országnak N (≤9) határátkelőhelye van. A ki- és a belépéseket nyilvántartják.

A határátkelőhelyek előző évi adatai a HATARx.BEy (y a határátkelőhely sorszáma) állományokban vannak az utasok határátlépésének sorrendjében: minden sorban egy-egy szóköz választja el az éven belüli napsorszámot, a KI vagy BE szót és az utas nevét (a név legfeljebb 40 tetszőleges karakterből áll).

Készíts programot, amely

A. a KULFOLDx.KI állományba írja azoknak a nevét, akik az adatok alapján az év végén külföldön tartózkodnak;

B. a SERTx:KI állományba írja azoknak a nevét, akik határsértést követtek el. (Határsértést az követett el, aki a nyilvántartás szerint azonos irányból egymás után egynél többször lépte át a határt.)

Példa: (N=2 esetére)

HATAR0.BE1	HATAR0.BE2	KULFOLD0.KI	SERT0.KI
22 KI Nagy Péter 23 BE Nagy Péter 36 BE Kovács Anna 44 KI Szép Tünde 170 BE Okos Lajos 192 BE Nagy Péter	38 KI Kovács Anna 167 KI Okos Lajos 178 KI Okos Lajos 188 KI Szép Tünde	Kovács Anna Szép Tünde Okos Lajos	Szép Tünde Nagy Péter

3. feladat: Kemping (17 pont)

A Napsugár Kemping K faházat üzemeltet az év 365 napján. Minden vendég ugyanannyi időre, pontosan M napra foglalhat le egy-egy faházat. A kemping a teljes évre előre összegyűjtötte az igényeket. Minden vendég igényként annak a napnak a sorszámát adta meg, amelytől kezdve (M napra) le akar foglalni egy faházat.

Írj programot, amely kiszámítja, hogy a kemping maximálisan hány vendéget fogadhat!

A KEMPING.BE állomány első sora a faházak számát (1≤K≤100) tartalmazza, második sora pedig az M (1≤M≤14) számot. A harmadik sorban az igények száma (1≤N≤ 1000), a további N sor mindegyikében pedig egy-egy igény van.

Az eredményt a KEMPING.KI állományba és a képernyőre kell írni!

Példa:

```
KEMPING0.BE     KEMPING0.KI
2               5
7
8
1
10
2
11
1
3
4
18
```

4. feladat: Mars-kutatás (17 pont)

A Mars felszínén különleges járművekkel kőzetmintákat gyűjtenek és juttatnak el az adóállomásra. A járművek csak négyzetrácsos útvonalon tudnak közlekedni, a kiindulási helytől csak dél vagy kelet felé és kizárólag sziklamentes terepen tudnak haladni. Térkép mutatja, hogy hol (melyik rácspontban) van (egy) kőzetminta, és hogy hol sziklás a terep. A térkép egy M*N-es mátrix, amely a különböző területfajtákat a következőképpen jelöli:

üres terület: 0 **kőzetmintás terület: 1** **sziklás terület: 2**

A járművek az (1,1) pontból indulnak, ez a térképen ábrázolt terep bal felső sarka. Az adóállomás az (M,N) pontban van (M a sorindex, N az oszlopindex).

Írj programot, amely kiszámítja az összegyűjthető kőzetminták számának maximumát, ha feltesszük, hogy elegendő számú jármű áll rendelkezésre!

A MARS.BE állomány első sorában az M és N értékek vannak egyetlen szóközzel elválasztva (1≤ M,N≤100). A következő M sor mindegyike a térkép egy sorát adja meg, azaz minden sorban N szám (0, 1 vagy 2) van, ahol a számokat ismét egyetlen szóköz választja el.

A feladat megoldásaként egyetlen számot kell írni a képernyőre és a MARS.KI állományba!

Példa:

MARS0.BE

```
10 12
0 1 2 1 0 0 1 0 2 0 0 0
0 1 0 0 0 2 0 0 0 0 0 0
0 0 0 1 0 0 0 0 2 0 0 0
2 0 0 0 0 2 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0
0 2 0 0 2 0 0 0 0 0 0 0
1 0 0 0 0 1 1 1 1 0 0 0
1 1 1 1 1 2 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 1 0 0
```

MARS0.KI

```
14
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Szálloda (14 pont)

Egy szállodalánc nyilvántartást vezet a szakácsairól és a cukrászairól. Ebben a szakképzettségi szint és az életkor szerint lehet keresni.

A SZAKACSx.BE és a CUKRASZx.BE szöveges állományok soronként egy-egy személyről három adatot tartalmaznak, egy-egy szóközzel elválasztva: a *szakképzettségi szintet* (`1` a legjobb, `10` a legrosszabb), az *életkort* (15 és 100 közötti egész) és a *nevet* (legfeljebb 40 karakter: az angol ábécé betűi, számjegyek és szóközök). Mindkét állományban legfeljebb 100 személy adatai vannak.

A megválaszolandó kérdések a KERDESx.BE állomány egy-egy sorában vannak `xxxx:yyyy:zzzz` alakban, ahol a mezőket : (kettőspont) választja el, és `xxxx` a szakképzettséget, `yyyy` a szakképzettségi szintre, `zzzz` pedig a korra vonatkozó követelményt adja meg.

`xxxx = SZAKACS` vagy `CUKRASZ` vagy `BARMILYEN`

`BARMILYEN` azt jelenti, hogy az illető *akár szakács, akár cukrász* lehet.

`yyyy = <N` vagy `>N` vagy `=N` vagy `<=N` vagy `>=N`

Az `yyyy` a szakképzettségi szint és az N egész közötti relációt írja elő.

`zzzz = MIN` vagy `MAX`

A `zzzz` azt adja meg, hogy az előző két feltételt kielégítő személyek közül a legfiatalabb vagy a legidősebb adatait kérjük-e.

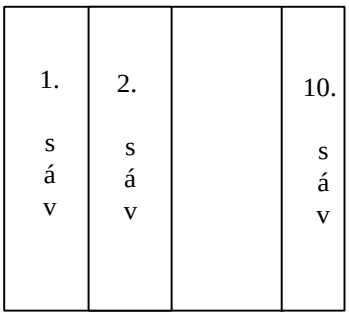
Az egyes kérdésekre válaszul kapott személyek nevét a VALASZx.KI állományba kell írni (soronként egy nevet)! Ha egy kérdésre nincs megfelelő személy, az adott sorba a `NINCS` szöveget kell írni! Ha egynél több van, az ábécé szerinti első nevet kell megadni!

Példa:

SZAKACS0.BE	CUKRASZ0.BE	KERDES0.BE	VALASZ0.KI
1 35 Nagy Lajos 4 49 Kiss Pal	1 37 Jo Istvan	SZAKACS:=4:MAX BARMILYEN:<2:MAX CUKRASZ:>=3:MIN	Kiss Pal Jo Istvan NINCS

2. feladat: Radar (20 pont)

Egy légvédelmi bázist olyan speciális radarral szereltek fel, amely egy 1000*1000-es területet ellenőriz, és észleli, ha ellenséges rakéta közeledik. A terület bal felső pontjának koordinátája (1,1). A bázis a (496,496) és (505,505) pontok által kijelölt négyzetben van.



A radar időegységenként egyetlen 100*1000-es sávot „lát”. (Az 1. sávval kezdi, majd a 2. sávval folytatja s.í.t.; a 10. sáv után újra az 1. következik). Csak olyan rakétát észlel, amelyik az éppen vizsgált sávban van.

A rakétákat a területen kívülről indítják. Egy-egy rakéta időegységenként mindig ugyanakkora utat tesz meg egyenes vonal mentén haladva, és mindkét koordinátája egész értékkel (legfeljebb 10-zel) változik meg. Ugyanazon a helyen több rakéta is lehet egy időben. A rakéta becsapódik, ha egy időegység kezdetén éppen a bázis területe fölött van. A rakéta repülés közben meg is semmisülhet (pl. lelőhetik).

A bázison állomásozó katonákat riasztani kell, ha olyan rakéta tart a bázis felé, amely a bázison csapódna be, és amelyet a radar már háromszor észlelt. Minden ilyen rakéta közeledését csak egyszer kell jelezni.

Írj programot, amely szükség esetén riasztja a katonákat!

A RAKETAx.BE állományban legfeljebb 1000 sor, minden sorban egy-egy észlelés három adata van, három egész szám, amelyeket egy-egy szóköz választ el: az időegység sorszáma, az észlelt rakéta sorindexe és oszlopindexe.

A program írja ki a képernyőre és a RAKETAx.KI állományba a riasztási adatokat! Minden sorba négy egész számot írjon, egy-egy szóközzel elválasztva: a riasztás idejét, a becsapódás várható idejét, sorindexét és oszlopindexét!

Példa:

RAKETA0.BE	RAKETA0.KI
1 1 1 7 506 5 11 21 21 17 526 35 21 41 41 27 546 65 31 61 61 37 566 95 41 81 81 48 588 128 51 101 101 61 121 121	21 249 497 497

3. feladat: Járműkölcsönzés (11 pont)

Egy közlekedési vállalat járműveket ad bérbe. Egyik különleges szállítójárművére nagy az igény, ezért a megrendeléseket a következő évre előre összegyűjtik. A megrendelést az A B számpárral adják meg, ami azt jelenti, hogy a megrendelő a járművet az év A-adik napjától a B-edik napjáig akarja bérbe venni. A vállalat a járművet a lehető legjobban akarja hasznosítani, ezért olyan megrendeléseket fogad el, amelyeket teljes egészükben úgy tud teljesíteni, hogy a jármű a lehető legtöbb napra ki legyen adva.

Írj programot, amely kiszámítja, hogy a megrendelések alapján a jármű a legjobb esetben hány napra lesz bérbe adva!

A JARMU.BE állomány tartalmazza a megrendeléseket. Az első sorban a megrendelések N száma van (1≤N≤1000). A következő N sor mindegyikében egy-egy megrendelés, .azaz egy-egy A B számpár van, egyetlen szóközzel elválasztva (1≤A≤B≤365).

A megoldást a képernyőre és a JARMU.KI állományba kell kiírni!

Példa:

```
JARMU0.BE          JARMU0.BE

9                  69
1 23
6 37
24 43
46 71
12 54
44 66
13 25
22 33
5 10
```

4. feladat: Robot (12 pont)

Egy üzemben a gyártást automatizálták. A szerszámgépek egy nagy gépcsarnokban négyzetrács mentén vannak elhelyezve. A műszak végén robotok gyűjtik össze a szerszámgépek gyártotta alkatrészeket. A robotok négyzetrács alakú pályán mozognak a szerszámgépek fölötti térben. A négyzetrács bal felső sarkából, az (1,1) pontból indulnak, és a jobb alsó sarokba viszik el az alkatrészeket. A robotokat úgy tervezték, hogy csak jobbra és „lefelé" haladhatnak.

Írj olyan programot, amely kiszámítja, hogy minimálisan hány robotot kell elindítani az összes alkatrész begyűjtéséhez, feltéve hogy minden robot tetszőleges számú alkatrészt szállíthat.

A feladat megoldásához rendelkezésre áll az üzemcsarnok térképe, amely egy 0 és 1 elemeket tartalmazó M*N-es mátrix (M a sorok, N az oszlopok száma). A 0 jelöli az üres helyet gépcsarnokban, az 1 pedig azt, hogy azon a helyen szerszámgép van.

A ROBOT.BE állomány első sorában az M és N értékek vannak, egy szóközzel elválasztva (1≤M, N≤100). A következő M sor mindegyike a térkép egy-egy sora, azaz minden sorban N szám (0 vagy 1) van, a számokat ismét egyetlen szóköz választja el.

A feladat megoldásaként egyetlen számot kell írni a ROBOT.KI állományba.

Példa:

```
ROBOT0.BE                    ROBOT0.KI

10 12                        5
0 1 1 1 0 0 1 0 1 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0 0
1 0 0 0 0 1 1 1 1 0 0 0
1 1 1 1 1 1 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 1 0 0
```

5. feladat: Fordítóprogram (18 pont)

Az alábbi programozási nyelven írt programokkal egy tár legfeljebb 1024 db 8 bites rekeszét adatokkal tölthetjük fel (az 1. rekesztől kezdve):

`Címke: x(DB)` *x*-et beírja a tár következő `DB` rekeszébe `(0≤x, DB≤255)`

`Címke: 'x'(DB)` az *x* karakter kódját beírja a tár következő `DB` rekeszébe `(0 ≤ DB ≤ 255)`

`Címke: "xxxxxx"` az `xxxxxx` karaktersorozat elemeinek kódját beírja a tár következő rekeszeibe

`Címke: <X>` az X címkének megfelelő 16 bites címet írja a tár következő két rekeszébe (előbb a kisebb helyi értékű részét, utána a nagyobbat)

`Címke: [X]` az X címkének megfelelő címen levő 8 bites értéket írja a tár következő rekeszébe

A `Címke:` maximum hat, az angol ábécé betűiből álló karaktersorozat, amelyet kettőspont követ. A `Címke:` el is maradhat. A számok és címkék belsejét kivéve mindenütt tetszőleges számú szóköz lehet.

Írj programot, amely beolvassa a lefordítandó, biztosan helyes programot a PROGRAMx.BE állományból, és kiírja a képernyőre és a PROGRAMx.KI állományba a fordítás eredményét (a tár rekeszeinek tartalmát)! A PROGRAMx.BE állományban minden utasítás külön sorban van.

Példa:

PROGRAM0.BE	PROGRAM0.KI

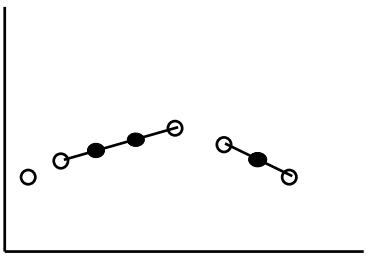
`A: 12 (2)`
`  'a'(3)`
`B: "bcd"`
`  [A]`
`  < A >`
`  <C  >`
`C: [B]`
	`12 12 97 97 97 98 99 100 12 1 0 14 0 98`

1998. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Méréskiegészítés (25 pont)

A Déli Sarkon az elmúlt N (≤100) napon minden délben megmérték a hőmérsékletet, kivéve a viharos napokat (ezek száma legfeljebb N/2). A feldolgozáshoz a hiányzó adatok becslésére is szükség van: A becsléshez feltételezzük, hogy a hiányzó napokon a hőmérséklet egyenletesen változott. Az ábrán azt az esetet mutatjuk, amikor a 2. és az 5. mérés között 2 hiányzott, a 6. és a 8. között pedig 1. A feketére festett karikák jelölik a becsült értékeket.



Ha a méréssorozat elején vagy a végén hiányzik mérés, akkor azok becsült értéke azonos legyen az első, illetve az utolsó érvényes mérés értékével

Készíts programot, amely beolvassa N értékét és az N mérési eredményt (tetszőleges valós számként, Celsius-fokban; a hiányzó értékeket –1000 jelöli.), majd kiírja a mérési eredményeket úgy, hogy a hiányzó méréseket a fenti módszer szerint becsült értékekkel pótolja!

Példa:

```
N=11
Mérések:
    -25.5,-20,-1000,-1000,-11,-10,-1000,-25,-22,-1000,-1000
Eredmény:
    -25.5,-20,  -17,  -14,-11,-10,-17.5,-25,-22,  -22,  -22
```

2. feladat: Karácsonyfa (25 pont)

A karácsonyi vásárban N helyen árulnak fenyőfát. Egy árus többféle fenyőt is árulhat, de azonos fajtájút csak egyféle áron adhat.

Készíts programot, amely először is beolvassa N értékét, majd pedig azt, hogy hányadik árus milyen fajta fenyőt árul, s annak méterét mennyiért adja! Ezek után a program írja ki, hogy:

A. melyik árusnál kapható a legolcsóbb fa,

B. a vásárban hányféle fenyő kapható,

C. melyik fajta fenyőt hány árus árulja,

D. az egyes fenyőfajták hol a legolcsóbbak.

Példa:

```
N=6
1. árus: ezüstfenyő, 3000
         erdeifenyő, 2000
2. árus: erdeifenyő, 1200
3. árus: ezüstfenyő, 2700
4. árus: feketefenyő, 1600
5. árus: ezüstfenyő, 3900
6. árus: feketefenyő, 1800
         erdeifenyő, 1600
```

```
A. A legolcsóbb fa a 2. árus-
nál kapható.
B. A fenyőfajták száma: 3
C. ezüstfenyő: 3 árus
   erdeifenyő: 3 árus
   feketefenyő: 2 árus
D. A legolcsóbb fenyőfajták:
    ezüstfenyő: 3. árus
    erdeifenyő: 2. árus
    feketefenyő: 4. árus
```

3. feladat: Karaktersorozat keresése (25 pont)

Egy szövegben adott karaktersorozatot az alábbi feltételek mellett kereshetünk:

A. a karaktersorozat csak teljes szó lehet, más szó részeként nem szabad megtalálni (a szavak az angol ábécé kis- és nagybetűiből állnak, és szóköz, írásjel vagy sorvég-jel határolja őket),

B. a karaktersorozat lehet szó része is;

C. nem szabad megkülönböztetni a nagy- és kisbetűket, azaz pl. `alma` és `ALmA` azonosnak számít;

D. meg kell különböztetni a nagy- és kisbetűket.

Készíts programot, amely beolvas egy karaktersorozatot, a keresési feltételek betűjelét (AC, AD, BC vagy BD) és egy szöveget, majd megadja, hogy a keresett karaktersorozat hányadik karaktertől kezdve található meg a szövegben (a karakterek számozását 1-től kezdjük)! Ha többször is megtalálható, akkor az összes előfordulás helyét meg kell adni! Ha egyszer sem található meg, akkor az eredmény 0 legyen!

Példa:

```
Keresett: is                    Eredmény: 4,11
Szöveg: Ez is, az is.
Keresési feltétel: BD
```

Kilencedik-tizedik osztályosok

1. feladat: Ütemező (50 pont)

Számítógépes szimulációkban események kezelésére használjuk az ESEM nyelvet. Az ESEM nyelvű *eseménygenerátorok* a generálandó események nevét és idejét definiálják (legfeljebb 200 eseménydefiníció):

```
EGbegin K *L       vagy       EGbegin K /L
  eseménydefiníció feltétel
  eseménydefiníció feltétel
  ...
EGend
```

A szimuláció a 0. időpillanatban indul és az időegysége 1. Az eseménygenerátorok egymásba ágyazhatók. Az eseménygenerátor ideje lokális, azaz minden benne szereplő érték a kezdőidejéhez képest relatív; működése a szimuláció K. lépésében kezdődik, és időegysége az előző szint időegységének L-szerese, ill. L-edrésze. (ESEM nyelven csak pozitív egész számok írhatók le, az osztás miatt a generált események időpontja azonban valós szám is lehet.)

Az *eseménydefiníciónak* három változata van:

- `n eseménynév` Az n. időpillanatban keletkezik az esemény.
- `n+k eseménynév` Az n. időpillanattól kezdve k időpillanatonként ismétlődik az esemény.
- `n+k<m eseménynév` Az n. időpillanattól kezdve az m. időpillanatig k időpillanatonként ismétlődik az esemény.

Az eseménydefiníció után, az `eseménynév`-től egy szóközzel elválasztva kétféle *feltételt* írhatunk elő (feltétel megadása nem kötelező):

- `<eseménynév2` Csak az `eseménynév2` esemény előtt keletkezhet az esemény.
- `>eseménynév2` Csak az `eseménynév2` esemény után keletkezhet az esemény.

Készíts programot, amely ESEM nyelven írt programokat hajt végre!

Az ESEMx.BE állomány első sorában a szimuláció T időtartama van (T>0), amelyet az ESEM nyelven megírt program követ (soronként 1 utasítás, a fent megadott formában).

Az ESEMx.KI állományba és a képernyőre a szimuláció eseményeit kell kiírni idő szerinti sorrendben a 0.-tól a T. időpontig. Minden sorban egy-egy esemény leírása legyen: az esemény időpontját és nevét kettőspont (:) válassza el egymástól. Ha az esemény időpontja nem egész szám, a kiíráskor csonkítani kell.

Példa:

ESEM0.BE	ESEM0.KI

20
EGbegin 0 *1
  10 egy
  2+3<16 ketto >egy
  2+3<10 harom
EGend
	2:harom
	5:harom
	8:harom
	10:egy
	11:ketto
	14:ketto

ESEMA.BE	ESEMA.KI

25
EGbegin 3 *2
  6 A
  Egbegin 0 *2
  3+2 B
  EGend
EGend
	15:A
	15:B
	23:B

2. feladat: Karaktersorozat keresése a Weben (25 pont)

Karaktersorozatot keresni összekapcsolt állományokban is lehet. Csak a kiinduló állomány nevét ismerjük, a többit (legfeljebb 200) a hivatkozásokból kell kiszedni.

Az **&**állománynév**&** alakú hivatkozások az állományban bárhol, akár ismételten is előfordulhatnak. (A kapcsolt állományokban természetesen további hivatkozások lehetnek.) A keresett karaktersorozatban sorvég-jel és **&**-jel nem lehet, és legfeljebb 250 karakterből állhat.

Karaktersorozatot az alábbi feltételek mellett kereshetünk:

A. a karaktersorozat csak teljes szó lehet, más szó részeként nem szabad megtalálni (a szavak az angol ábécé kis- és nagybetűiből állnak, és szóköz, írásjel vagy sorvég-jel határolja őket),

B. a karaktersorozat lehet szó része is;

C. nem szabad megkülönböztetni a nagy- és kisbetűket, azaz pl. alma és ALmA azonosnak számít;

D. meg kell különböztetni a nagy- és kisbetűket.

Készíts programot, amely megadja, hogy a keresett karaktersorozat mely állományok hányadik karakterétől kezdve található meg (sem a teljes hivatkozás, sem a sorvég-jel **nem** számít karakternek)! Ha ugyanaz a karaktersorozat többször is megtalálható, akkor mindegyik előfordulás helyét meg kell adni.

A WEBx.BE állomány első sora a kiinduló állomány nevét, a második a keresési feltételek betűjelét (AC, AD, BC vagy BD), további tetszőleges számú sora pedig a keresendő karaktersorozatokat tartalmazza.

A WEBx.KI állományba és a képernyőre az alábbiakat kell írni:

Minden keresés eredménye annyi sor, ahány állományban megtalálható a keresett karaktersorozat. A sor elején az állomány neve legyen, majd egy kettőspont (:), utána pedig a karaktersorszámok egy-egy szóközzel elválasztva. Az egyes keresési eredmények között egy-egy üres sort kell hagyni.

Példa:

WEB0.BE:	**ELSO.TXT:**	**WEB0.KI:**
ELSO.TXT BC xyz zzz	xyz zzz Xyz, qwert &Masodik.txt& zzz **MASODIK.TXT:** UVWXYZ	ELSO.TXT: 1 8 MASODIK.TXT: 4 ELSO.TXT: 5 18

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Terv (50 pont)

Egy nagyszabású építkezéshez részletes tervet készítettek, amelyben többek között azt is megadják, hogy egy-egy munka hány napig tart. A munkák sorrendje nem tetszőleges. A tervben a sorrendet

a b feltételpárok írják elő. Az a b feltételpár azt jelenti, hogy a b munkát csak az a munka befejezése után lehet elkezdeni. Az építkezés az 1. napon kezdődik.

Írj programot az alábbi részfeladatok megoldására!

A. Számítsd ki, hogy a terv végrehajtásához minimálisan hány nap szükséges!

B. Add meg az egyes munkák legkorábbi kezdési idejét a számuk sorrendjében! (Ez az a legkorábbi időpont, amikor az adott munkát el lehet kezdeni, mert az összes korábbi, ugyancsak a lehető legkorábban elkezdett munka már befejeződött.)

C. Add meg a munkák legkésőbbi kezdési idejét a számuk sorrendjében! (Ez az a legkésőbbi időpont, amikor az adott munkát el kell kezdeni ahhoz, hogy ne késleltesse az építkezés befejezését.)

D. Adj meg egy ún. kritikus utat! (Kritikus útnak nevezzük olyan munkák egy sorozatát, amelyeket az adott sorrendben lehet elvégezni, és a munkák összideje megegyezik a terv végrehajtásához minimálisan szükséges idővel, továbbá minden egyes munka legkorábbi és legkésőbbi kezdési ideje ugyanaz.)

E. Add meg, hogy minimum hány vállalkozóval kell szerződést kötni a munkák elvégzésére, ha egy vállalkozó egy időben csak egy munkán tud dolgozni és ha minden munka a lehető legkorábban kezdődik!

F. Add meg, hogy minimálisan mennyi az összes állási idő! (Az állási idő abból adódik, hogy egy vállalkozó két, időben egymást követő munka elvégzése között várakozásra kényszerülhet, ha a soron következő munkát még nem kezdheti el.) Feltesszük, hogy minden munka a lehető legkorábban kezdődik, és a kivitelezők a lehető legkevesebb vállalkozót alkalmazzák.

A TERVx.BE állomány első sora a munkák számát ($1 \le N \le 100$) tartalmazza. A második sorban N db pozitív egész szám van, ahol az i-edik szám a sorban az i-edik munka végrehajtásához szükséges napok száma (≤ 100). A harmadik sorban a feltételpárok száma ($0 \le M \le 1000$) van. A következő M sor mindegyike egy-egy a b számpárt tartalmaz ($1 \le a, b \le N$); ami azt jelenti, hogy a b munkát csak az a munka befejezése után lehet elkezdeni.

A TERVx.KI szöveges állományba és a képernyőre hat sorba kell írni az A.–F. részfeladatok megoldását! Ha valamelyik részfeladat megoldása hiányzik, akkor a megfelelő sor legyen üres! A B., C. és D. részfeladatok esetén (2., 3. és 4. sor) a számsorozatok elemeit egy-egy szóköz válassza el!

Példa:

TERV0.BE	TERV0.KI
5 1 2 1 3 2 5 1 2 2 5 1 3 3 4 2 4	6 1 2 2 4 4 1 2 3 4 5 1 2 4 2 1

2. feladat: Karaktersorozat keresése a Weben (25 pont)

Karaktersorozatot keresni összekapcsolt állományokban is lehet. Csak a kiinduló állomány nevét ismerjük, a többit (legfeljebb 200) a hivatkozásokból kell kiszedni.

Az **&**állománynév**&** alakú hivatkozások az állományban bárhol, akár ismételten is előfordulhatnak. (A kapcsolt állományokban természetesen további hivatkozások lehetnek.) A keresett karaktersorozatban sorvég-jel és **&**-jel nem lehet, és legfeljebb 250 karakterből állhat.

Karaktersorozatot az alábbi feltételek mellett kereshetünk:

A. a karaktersorozat csak teljes szó lehet, más szó részeként nem szabad megtalálni (a szavak az angol ábécé kis- és nagybetűiből állnak, és szóköz, írásjel vagy sorvég-jel határolja őket),

B. a karaktersorozat lehet szó része is;

C. nem szabad megkülönböztetni a nagy- és kisbetűket, azaz pl. `alma` és `ALmA` azonosnak számít;

D. meg kell különböztetni a nagy- és kisbetűket.

Készíts programot, amely megadja, hogy a keresett karaktersorozat mely állományok hányadik karakterétől kezdve található meg (sem a teljes hivatkozás, sem a sorvég-jel **nem** számít karakternek)! Ha ugyanaz a karaktersorozat többször is megtalálható, akkor mindegyik előfordulás helyét meg kell adni!

A WEBx.BE állomány első sora a kiinduló állomány nevét, a második a keresési feltételek betűjelét (AC, AD, BC vagy BD), további tetszőleges számú sora pedig a keresendő karaktersorozatokat tartalmazza.

A WEBx.KI állományba és a képernyőre az alábbiakat kell írni!

Minden keresés eredménye annyi sor, ahány állományban megtalálható a keresett karaktersorozat. A sor elején az állomány neve legyen, majd egy kettőspont (:), utána pedig a karaktersorszámok egy-egy szóközzel elválasztva! Az egyes keresési eredmények között egy-egy üres sort kell hagyni!

Példa:

WEB0.BE:	**ELSO.TXT:**	**WEB0.KI:**
ELSO.TXT BC xyz zzz	xyz zzz Xyz, qwert &Masodik.txt& zzz **MASODIK.TXT:** UVWXYZ	ELSO.TXT: 1 8 MASODIK.TXT: 4 ELSO.TXT: 5 18

3. feladat: Kommandó (25 pont)

Egy térképrészleten minden ország be van festve valamilyen színnel. Egy kommandónak el kell jutnia az egyik országból egy másikba. A kommandósoknak, hogy ne ismerjék fel őket, minden országban olyan színű trikót kell hordaniuk, amilyen az adott ország színe a térképen. Határozd meg azt a legrövidebb útvonalat, amelynek a bejárásához a lehető legkevesebbszer kell trikót cserélni.

A KOMx.BE állomány első sorában az országok száma ($2 \le N \le 100$) van:. A következő N sorban ismeretlen számú egész szám található: az egyes országok és szomszédjaik országkódja. A következő N sorban az egyes országok színkódja található (ugyancsak egész számok, az országkódok növekvő sorrendjében). Az utolsó sorban 2 egész szám azonosítja a kiindulási országot és a célországot.

A KOMx.KI állományba és a képernyőre a felhasználandó trikók színkódját és az útvonalon fekvő országok országkódját kell kiírni! Ha több azonos értékű megoldás van, elég egyet megadni.

Példa:

KOM0.BE	KOM0.KI:
6 1 2 3 4 2 1 3 6 3 1 2 4 5 6 4 1 3 5 3 6 6 2 3 5 1 3 2 2 3 1 1 5	1 3 1 2 6 5

A verseny végeredménye:

I. kategória

1. Micskó Viktor	Lengyel József Gimnázium, Oroszlány
2. Siroki László	Fazekas Mihály Gimnázium, Debrecen
Pszota Zsolt	Petőfi Sándor Általános Iskola, Vác
Szatmári Zsolt	Lehel Vezér Gimnázium, Jászberény
5. Balogh János	Kinizsi Lakótelepi Általános Iskola, Kaposvár
6. Sztupák Szilárd	Hermann Ottó Gimnázium, Miskolc
7. Béky Bence	Fazekas Mihály Gimnázium, Budapest
8. Juhász Sándor	Bocskai István Általános Iskola, Hajdúböszörmény
Szeredi Dániel	Veres Péter Gimnázium, Budapest
10. Hargitai Gábor	Bolyai János Gimnázium, Ócsa

II. kategória

1. Rokob András	Földes Ferenc Gimnázium, Miskolc
2. Sáfár Szilveszter	Ságvári Endre Gimnázium, Szeged
3. Varga Kornél	Földes Ferenc Gimnázium, Miskolc
4. Soós István	Kanizsai Dorottya Gimnázium, Szombathely
5. Ritter Ádám	Fazekas Mihály Gimnázium, Budapest
6. Csillag Kristóf	Karacs Ferenc Gimnázium, Püspökladány
7. Nagy Zsombor	Eötvös József Gimnázium, Tata
Csirmaz Előd	Fazekas Mihály Gimnázium, Budapest
9. Flach Attila	Ságvári Endre Gimnázium, Szeged
Pollák Gergely	Ságvári Endre Gimnázium, Szeged

III. kategória

1. Szabó Viktor	Leövey Klára Gimnázium, Budapest
2. Kőműves Balázs	Fazekas Mihály Gimnázium, Budapest

3. Marhefka István	Avasi Gimnázium, Miskolc
4. Balogh András	Lovassy László Gimnázium, Veszprém
5. Nagy András	Leőwey Klára Gimnázium, Pécs
Németh András	Fazekas Mihály Gimnázium, Budapest
7. Felföldi Zsolt	Fazekas Mihály Gimnázium, Budapest
8. Förhécz András	Teleki Blanka Gimnázium, Székesfehérvár
Pintér Lóránt	Táncsics Mihály Gimnázium, Kaposvár
10. Kopiás Péter	Neumann János Szakközépiskola, Budapest
Varga Péter	Premontrei Rendi Gimnázium, Szombathely

1999. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mit rajzol (25 pont)

Mit rajzol az alábbi Logo eljárás

A. `:n=5`, `:szög=720`, `:h=100`, illetve

B. `:n=11`, `:szög=1650`, `:h=100` esetén?

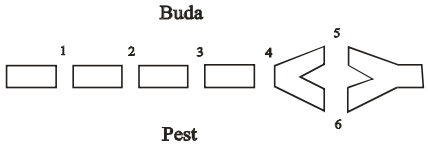
```
tanuld miez :n :szög :h
 ismétlés :n [előre :h balra ( :szög-360 )/:n
              előre :h jobbra :szög/:n]
vége
```

Rajzold le a két alakzatot, s add meg a szögeiket!

C. Milyen alakzatot rajzol a program az A részfeladatbeli paraméterezés esetén, ha a `balra ( :szög-360 )/:n` *után* beillesztjük a `hátra :h előre :h` utasításokat?

2. feladat: Robot (25 pont)

A budapesti hidak Budát és Pestet, a Margitszigetet és Budát, illetve a Margitszigetet és Pestet kötik össze. A hidakat az ábrán látható módon számozzuk.



Egy robotot készítünk, amelyik `Buda`, `Pest`, illetve a `Margitsziget` között közlekedik. A robot kezdetben `Budán` van. A robotot egy 1 és 6 közötti számokból álló számsorozattal utasítjuk arra, hogy az adott sorrendben haladjon át a hidakon. Ha hibás utasítást kap, megáll.

Példa:

> Az 1,4,5,6 sorozatra a robot Budáról indulva a következő helyekre lép: (1 hatására) Pest, (4 hatására) Buda, (5 hatására) Margitsziget, (6 hatására) Pest.

A. Az (1,3,5,5,1,6) sorozat hatására hol lesz a robot a folyamat végén?

B. Az alábbi sorozatok közül melyek a hibás utasítások és miért?

B1. 1,1,1,1,5 B2. 1,1,1,5 B3. 3,1,5,5,6 B4. 2,1,5,1

C. Sok olyan sorozat van, amelyiknek ugyanaz lesz a végeredménye. Fogalmazd meg azokat az egyszerűsítési szabályokat, amelyekkel a helyes utasítások úgy tehetők a lehető legrövidebbé, hogy a végeredmény változatlan marad!

D. Melyek azok a legrövidebb utasítások, amelyekre a robot végállomása `Buda`, `Pest`, illetve a `Margitsziget`?

3. feladat: Cserélgetés (25 pont)

Az alábbi algoritmus az `N` elemű `A` vektor alapján határozza meg `D` és `M`, valamint az `Y` vektor értékét.

```
Valami(A,N):
  D:=1; M:=A(1)
  Ciklus I=2-től N-ig
     Ha M<A(I) akkor Y(D):=M; D:=D+1; M:=A(I)
     A(I):=M
  Ciklus vége
  Y(D):=M
Eljárás vége.
```

A. Fogalmazd meg, hogy mi lesz az eljárás hatására M, D és Y értéke?

B. Fogalmazd meg, hogyan alakítja át az eljárás az A vektort?

C. Mi a feltétele annak, hogy az eljárás hatására az A vektor összes eleme egyforma legyen?

D. Mi a feltétele annak, hogy az eljárás hatására az A vektor elemei ne változzanak meg?

4. feladat: Véletlen (25 pont)

Az N és M természetes számokkal (0<N<M) paraméterezett Valami eljárás előállít egy A vektort:

```
Valami(M,N,A):
  Ciklus i=1-től M-ig
     k:=Véletlen(i)
     Ha i≤N akkor A(i):=A(k)
     Ha k≤N akkor A(k):=i
  Ciklus vége
  A(Véletlen(N)):=0
Eljárás vége.
```

A Véletlen(i) függvényeljárás egyenlő eséllyel állít elő 1 és i közötti egész számokat, beleértve az 1-et és az i-t is.

A. Milyen értékek kerülnek az A tömbbe? Milyen kapcsolat van köztük? Milyen a sorrendjük?

B. Ha M=9, N=6, akkor a véletlen értékeknek milyeneknek kell lenniük ahhoz, hogy az eredmény A=(0,5,6,7,8,9) legyen?

C. Ha M=9, N=6, akkor a véletlen értékeknek milyeneknek kell lenniük ahhoz, hogy az eredmény A=(0,1,2,3,4,5) legyen?

Kilencedik-tizedik osztályosok

1. feladat: Halmazok metszete (16 pont)

Az alábbi algoritmus két *halmaz* metszetét határozza meg. A két halmazt az N elemszámú A és az M elemszámú B tömbben tároljuk.

```
Metszet:
  K:=0
  Ciklus I=1-től N-ig
     J:=1
     Ciklus amíg J≤M és A(I)≠B(J)
        J:=J+1                    (*)
     Ciklus vége
     Ha J≤M akkor K:=K+1; C(K):=A(I)
  Ciklus vége
Eljárás vége.
```

A. *Adott* N és M esetén (N≤M) milyen adatokra a *leggyorsabb* az algoritmus, azaz a (*)-gal jelölt sort milyen A és B vektorra hajtja végre a *legkevesebbszer*? A (*)-gal jelölt sort ebben az esetben hányszor hajtja végre?

B. *Adott* N és M esetén (N≤M) milyen adatokra a *leglassúbb* az algoritmus, azaz a (*)-gal jelölt sort milyen A és B vektorra hajtja végre a *legtöbbször*? A (*)-gal jelölt sort ebben az esetben hányszor hajtja végre?

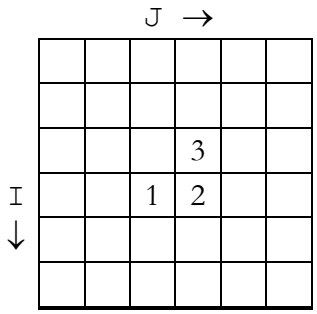
2. feladat: Festegetés (15 pont)

Az alábbi algoritmus a képernyő pontjait festi ki az (I,J) pontból kiindulva, a DB tömb feltöltésétől függő sorrendben.

```
Fest(I,J):
  K:=1; L:=1
  Ciklus amíg KépernyőnBelül(I,J)
    Pontrajzolás(I,J)
    I:=I+T(L,1); J:=J+T(L,2); DB(K):=DB(K)-1
    Ha DB(K)=0 akkor K:=K+1; L:=L mod 4 + 1
  Ciklus vége
Eljárás vége.
```

A `KépernyőnBelül(I,J)` függvényeljárás igaz értéket ad eredményül, ha az `(I,J)` pont a képernyőn belül van. A `Pontrajzolás(I,J)` eljárás kivilágítja az `(I,J)` pontot. A négyszer két elemű `T` tömb elemeinek tartalma:

`T(1,1)=0, T(1,2)=1, T(2,1)=-1, T(2,2)=0,`
`T(3,1)=0, T(3,2)=-1, T(4,1)=1, T(4,2)=0.`



Milyen sorrendben festi ki az algoritmus *az ábrán látható* terület pontjait, ha a `DB` vektort az alábbiak szerint töltjük ki? Másold le az ábrát 3-szor, és írd bele a további lépések sorszámát!

A. `DB=1,1,2,2,3,3,4,4,5,5,6,...`

B. `DB=1,2,3,4,5,6,...`

C. `DB=1,2,2,3,4,4,5,6,...`

3. feladat: Egyesítés (20 pont)

Az alábbi algoritmus két rendezett vektor (`A`, `B`) elemeiből állít elő egy újabb rendezett vektort, amelyben azok az elemek szerepelnek, amelyek legalább az egyikben előfordulnak. Az eredmény *minden elemének különbözőnek* kell lennie.

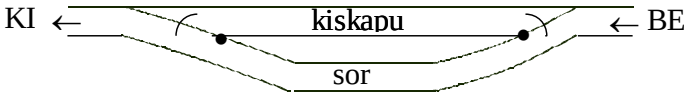
```
Összefuttatás(A,N,B,M,C,K):
  I:=1; J:=1; K:=0
  Ciklus amíg I≤N és J≤M
     K:=K+1
     Ha A(I)<B(J) akkor C(K):=A(I); I:=I+1
     különben ha A(I)=B(J)akkor C(K):=A(I); I:=I+1; J:=J+1
     különben C(K):=B(J); J:=J+1
  Ciklus vége
Eljárás vége.
```

A. Milyen feltételeknek kell teljesülniük `A`-ra és `B`-re az eljárás helyes működéséhez?

B. Milyen hibát okoz az eredményben, ha ezek a feltételek nem teljesülnek?

4. feladat: Kiskapus sor (25 pont)

A *kiskapus sor* olyan adatszerkezet, amelyen háromféle műveletet lehet végrehajtani: `SORBA` a BE bemenetre érkező jelet beteszi a sor végére, `SORBÓL` kirakja a következő jelet a sor elejéről a KI kimenetre, `ÁT` pedig átengedi a BE bemenetre érkező jelet a KI kimenetre. A kiskapus sort illusztrálja az ábra.



Feltesszük, hogy a BE bemenetre az 1,2,3,4,5 jelsorozat érkezik.

Példa:

Az 5,1,2,3,4 sorozat előállítható a `SORBA`, `SORBA`, `SORBA`, `SORBA`, `ÁT`, `SORBÓL`, `SORBÓL`, `SORBÓL`, `SORBÓL` műveletsorozattal.

A. Elő lehet-e állítani az alábbi sorozatokat? Amelyiket nem, azt meddig lehet? Amelyiket igen, azt hogyan lehet minimális számú művelettel előállítani?

A1. 3,1,4,2,5 A2. 2,4,3,1,5 A3. 5,1,2,4,3 A4. 2,3,1,5,4

B. Fogalmazd meg, hogy milyen sorrendű lehet az eredménysorozat!

5. feladat: Megszakítások (24 pont)

Egy K szintű megszakítási rendszerben a magasabb szintű megszakításkérés megszakítja az alacsonyabb szintű kiszolgálását, az éppen kiszolgáltnál (az *aktuálisnál)* nem magasabb szintű megszakításkéréseknek viszont várakozniuk kell. (A megszakítási szinteket természetes számokkal kódoljuk, a *legalacsonyabb szint* az 1-es.) Mindegyik szinthez egy-egy *várakozósor* (SOR) tartozik, amelyben a *még fel nem dolgozott* megszakításkérések legfontosabb adatait tároljuk. Feltesszük, hogy minden megszakításkérés kiszolgálása azonos ideig, pontosan 10 időegységig tart.

A megszakítás-kiszolgáló olyan objektum, amelynek AKT nevű mezőjében a *legutóbb kiszolgált megszakítás* szintjét tároljuk, s a következő három eljárást képes végrehajtani:

```
Init:                              [végrehajtandó a rendszer indításakor]
  AKT:=K; VAN:=hamis
Eljárás vége.

Megszakításkérés(P,NÉV):           [végrehajtandó megszakításkéréskor]
  R.NÉV:=NÉV; R.IDŐ:=0; Sorba(P,R)
  VAN:=igaz
Eljárás vége.

Kiszolgálás:          [végrehajtandó VAN=igaz esetén]
  Ciklus amíg AKT≥1 és Üres(AKT)
    AKT:=AKT-1
  Ciklus vége
  Ki: Első(AKT).NÉV
  R:=Első(AKT); R.IDŐ:=R.IDŐ+1; ElsőtMódosít(AKT,R)
Eljárás vége.
```

A Sorba(P,R) eljárás a P szintű megszakításkérést leíró (NÉV és IDŐ mezőkből álló) rekordot bejegyzi a P megszakítási szint várakozósorának végére. A Sorból(P) eljárás törli a P szint várakozósorából a soron következő megszakításkérést. Az Első(P) függvényeljárás egy rekordot ad eredményül: a P szint várakozósorában tárolt megszakításkérések közül az elsőt leíró rekordot. Az ElsőtMódosít(P,R) eljárás a P szint várakozósorában levő *első* rekordot írja fölül az R rekorddal. Az Üres(P) függvényeljárás igaz eredményt ad, ha a P szint várakozósorában már nincs több kiszolgálásra váró megszakításkérés.

Milyen hibák vannak a Megszakításkérés és a Kiszolgálás eljárásban, s hogyan lehet ezeket kijavítani?

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Időszerű (15 pont)

Az alábbi algoritmus két rendezett állomány (A és B) elemeiből állít elő egy újabb *rendezett* állományt (C). A B állomány minden egyes rekordja egy-egy parancs, amelyet az A állományon kell végrehajtani. A parancsok háromfélék lehetnek:

törlés: az A állomány adott kulcsú rekordját ne rakd be a C állományba!

javítás: az A adott kulcsú rekordja helyett a B-beli rekordot rakd be a C-be!

beszúrás: a parancsban megadott B-beli rekordot rakd be a C-be!

A két állomány rekordstruktúrája:

```
A=Rekord(kulcs: Szöveg, egyéb: Szöveg)

B=Rekord(fajta: (beszúrás,törlés,javítás),
         kulcs: Szöveg, egyéb: Szöveg)
```

A B-beli parancsokat az alábbi Időszerűsítés eljárás hajtja végre (a Vége(f) függvény IGAZ értékű, ha az f állományból az utolsó olvasás sikertelen volt):

```
Időszerűsítés(A,B,C):
  Olvas(A,X); Olvas(B,Y)
  Ciklus amíg nem Vége(A) és nem Vége(B)
     Ha X.kulcs < Y.kulcs akkor Ír(C,X); Olvas(A,X)
     különben ha X.kulcs = Y.kulcs akkor Valami1
     különben Valami2
  Ciklus vége
Eljárás vége.

Valami1:
  Ha Y.fajta = javítás akkor X.egyéb:=Y.egyéb; Ír(C,X)
                             Olvas(A,X); Olvas(B,Y)
  Elágazás vége
Eljárás vége.

Valami2:
  Ír(C,(Y.kulcs,Y.egyéb)); Olvas(B,Y)
Eljárás vége.
```

Feltehetjük, hogy az A és a B állomány mindig helyes: rendezett és nincsenek benne ismétlődő kulcsú rekordok. Ismétlődő kulcsú rekordok a C állományba sem írhatók!

A. Egy bizonyos esetben a fenti algoritmus akkor is hibásan működik, ha a B állományban csupa helyes parancs van.

A1. Milyen esetben?

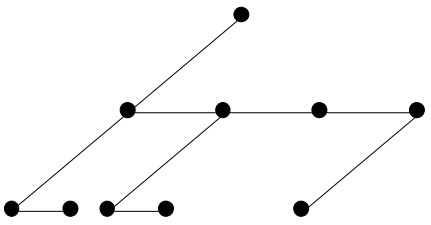
A2. Mi a rossz az eredményben?

A3. Fogalmazd meg, mit kell tenni, azaz hogyan lehet kijavítani a hibát!

B. Milyen hibás, a definíciónak nem megfelelő parancsok lehetnek a B állományban?

2. feladat: Családfa (15 pont)

Egy családfa az első gyereket a szülőtől balra, a többi gyereket az első gyerektől jobbra ábrázolja. A családfán a következő műveleteket értelmezzük:



Balra(f): az f-nek az a részcsaládfája, amelynek gyökéreleme az f gyökérelemének első gyereke;

Jobbra(f): az f-nek az a részcsaládfája, amelynek gyökéreleme az f gyökérelemének első testvére;

Siker: igaz, ha a fenti két művelet közül a legutóbb végrehajtott sikeresen ért véget.

Példa: a legfiatalabb olyan elsőszülött, akinek minden őse elsőszülött

```
Elsőszülött(f):
  Ciklus
    g:=f; f:=Balra(f)
  amíg Siker
  Ciklus vége
  Elsőszülött:=g
Eljárás vége.
```

A. Mi az eredménye az alábbi algoritmusoknak?

A1.

```
Egyik(f):
  f:=Balra(f); db:=0
  Ciklus amíg Siker
     f:=Jobbra(f):
     db:=db+1
  Ciklus vége
  Egyik:=db
Eljárás vége.
```

A2.

```
Másik(f):
  g:=Balra(f)
  Ha Siker akkor db:=Másik(g)
        különben db:=1
  f:=Jobbra(f)
  Ciklus amíg Siker
    db:=db+Másik(f)
    f:=Jobbra(f)
  Ciklus vége
  Másik:=db
Eljárás vége.
```

B. Írj olyan algoritmust, amely meghatározza a családfa gyökéreleméből ábrázolt szülő legkisebb gyerekét! (Feltesszük, hogy van gyereke.)

3. feladat: Logikusan (17 pont)

Az alábbi, PROLOG-szerű nyelven felírt tények azt írják le, hogy ki mikor született:

```
Született("Éva",1959).
Született("András",1943).
Született("Anna",1959).
Született("István",1961).
```

Azt, hogy X és Y ugyanabban az évben született, így írhatjuk le:

```
Egyidõsek(X,Y) ha Született(X,E) és Született(Y,E)és X<>Y.
```

(A fenti tényállítások alapján `Egyidősek(X,Y)` teljesül, és az egyik lehetséges válasz: `X=Éva` és `Y=Anna`.)

Mit ír le az alábbi *öt* szabály, azaz milyen paraméterekre teljesülnek, továbbá mi lesz az értéke X-nek, illetve Y-nak a fenti tényállítások mellett? (Az A., B. és C. részfeladatok függetlenek egymástól.)

A.
```
MilyenA(X) ha Született(X,E) és Született(Y,F) és E<F.
```

B.
```
MilyenB(X) ha Született(X,E) és nem(Valami(E)).
Valami(E) ha Született(Y,F) és E>F.
```

C.
```
MilyenC(X,Y) ha Született(X,E) és Született(Y,F) és
                  X<>Y és nem(Ilyen(abs(E-F))).
Ilyen(G) ha Született(P,A) és Született(Q,B) és
            abs(A-B)<G.
```

4. feladat: Kitalálós (15 pont)

Az alábbi algoritmusok 16 bites, előjel nélküli egész számokkal bitenkénti műveleteket végeznek. A `XOR` *kizáró vagy* művelet, az `SHL` *bitenkénti balra tolás,* jobbról 0 jön be, a legértékesebb bit elveszik.

```
Mitcsinál(A,B):                Találdki(A,B):
  Ciklus amíg B≠0                Ciklus amíg B≠0
    A:=A XOR B                     A:=A XOR B
    B:=SHL((A XOR B) AND B)        B:=SHL((NOT(A XOR B)) AND B)
  Ciklus vége                    Ciklus vége
  Mitcsinál:=A                   Találdki:=A
Függvény vége.                 Függvény vége.
```

A. Mi a két függvény feladata?

B. Mi a `B` változó szerepe a ciklus belsejében?

C. Maximum hányszor hajtja végre a ciklus magját a két függvényeljárás?

5. feladat: Megszakítások (21 pont)

Egy `K` szintű megszakítási rendszerben a magasabb szintű megszakításkérés megszakítja az alacsonyabb szintű kiszolgálását, az éppen kiszolgáltnál *(az aktuálisnál)* nem magasabb szintű megszakításkéréseknek viszont várakozniuk kell. (A megszakítási szinteket természetes számokkal kódoljuk, a *legalacsonyabb szint* az 1-es.) Mindegyik szinthez egy-egy *várakozósor* (`SOR`) tartozik, amelyben a *még fel nem dolgozott* megszakításkérések legfontosabb adatait tároljuk. Feltesszük, hogy minden megszakításkérés kiszolgálása azonos ideig, pontosan 10 időegységig tart.

A megszakítás-kiszolgáló olyan objektum, amelynek `AKT` nevű mezőjében a *legutóbb kiszolgált megszakítás* szintjét tároljuk, s a következő három eljárást képes végrehajtani:

```
Init:                          [végrehajtandó a rendszer indításakor]
  AKT:=K; VAN:=hamis
Eljárás vége.

Megszakításkérés(P,NÉV):       [végrehajtandó megszakításkéréskor]
  R.NÉV:=NÉV; R.IDÕ:=0; Sorba(P,R)
  VAN:=igaz
Eljárás vége.

Kiszolgálás:                   [végrehajtandó VAN=igaz esetén]
  Ciklus amíg AKT≥1 és Üres(AKT)
    AKT:=AKT-1
  Ciklus vége
  Ki: Első(AKT).NÉV
  R:=Első(AKT); R.IDÕ:=R.IDÕ+1; ElsőtMódosít(AKT,R)
Eljárás vége.
```

A `Sorba(P,R)` eljárás a `P` szintű megszakításkérést leíró (`NÉV` és `IDŐ` mezőkből álló) rekordot bejegyzi a `P` megszakítási szint várakozósorának végére. A `Sorból(P)` eljárás törli a `P` szint várakozósorából a soron következő megszakításkérést. Az `Első(P)` függvényeljárás egy rekordot ad eredményül: a `P` szint várakozósorában tárolt megszakításkérések közül az elsőt leíró rekordot. Az `ElsőtMódosít(P,R)` eljárás a `P` szint várakozósorában levő *első* rekordot írja fölül az `R` rekorddal. Az `Üres(P)` függvényeljárás igaz eredményt ad, ha a `P` szint várakozósorában már nincs több kiszolgálásra váró megszakításkérés.

Milyen hibák vannak a `Megszakításkérés` és a `Kiszolgálás` eljárásban, s hogyan lehet ezeket kijavítani?

6. feladat: Gyorskereső (17 pont)

Az `U` változó értéke akkor lesz igaz az alábbi algoritmusban, ha az `MI[1..M]` minta előfordul az `S[1..N]` szövegben. A karaktereket 1-től 255-ig kódoljuk. Különleges jelentése van a 0 kódú karakternek.

```
Kereső:
  Ciklus J=0-tól 255-ig
     SH(J):=M+1
  Ciklus vége
  Ciklus J=1-től M-ig
     SH(MI(J)):=M+1-J
  Ciklus vége
  K:=0; L:=1; S(N+1):=0
  Ciklus amíg K<M és L≤N-M+1
     Ha S(K+L)=MI(K+1) akkor K:=K+1
                       különben L:=L+SH(S(L+M)); K:=0
  Ciklus vége
  U:=(K=M)
Eljárás vége.
```

A. Mi van az SH vektorban a *második* ciklus lefutása után?

B. Rögzített N és M, továbbá a mintát nem tartalmazó szöveg esetén milyen mintára és szövegre *minimális* a *harmadik* ciklus lépésszáma? Mennyi ez a lépésszám?

C. Rögzített N és M, továbbá a mintát nem tartalmazó szöveg esetén milyen mintára és szövegre *maximális* a *harmadik* ciklus lépésszáma? Mennyi ez a lépésszám?

1999. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Szimmetrikusan (15 pont)

Ballagásra az osztályterem egyik falát szimmetrikusan próbálták díszíteni egymás mellé elhelyezett virágokkal. Mindkét szélén ugyanolyan színű virágnak kellene lennie, eggyel beljebb mindkét oldalról újra ugyanolyannak, ... Készíts programot, amely beolvassa a virágok számát (1≤N≤100), majd N darab virágszínt, majd kiírja, hogy kívülről befelé haladva a közepéig melyik helyeken tévesztették el a szimmetrikus díszítést! Ha a virágszín sorozat szimmetrikus, akkor a TÖKÉLETES szöveget kell kiírni.

Példa:

```
Bemenet:
  7
  PIROS SÁRGA FEHÉR FEHÉR SÁRGA SÁRGA KÉK
Kimenet:
  1 3
```

2. feladat: Naptár és óra (29 pont)

Egy számítógépen a dátumot és az időt az alábbiakkal adják meg:

Évszám, hónapnév, napsorszám, a nap a hét milyen napja, óra, perc, másodperc.

Készíts programot, amely beolvas egy dátum és időmegadást, majd billentyű lenyomásonként az 1 másodperccel megnövelt értéket írja ki, mindaddig, amíg a V billentyűt nem nyomjuk le.

Példa:

1999. január 31. vasárnap 23 óra 59 perc 58 másodperc

1999. január 31. vasárnap 23 óra 59 perc 59 másodperc

1999. február 1. hétfő 0 óra 0 perc 0 másodperc

1999. február 1. hétfő 0 óra 0 perc 1 másodperc

...

3. feladat: Mozijegyek (31 pont)

A piripócsi általános iskolában M (1≤M≤20) osztály van. Az iskola kapott N (1≤N≤ 200) mozijegyet, amelyet igazságosan szeretnének elosztani az osztályok között. Teljesen igazságos akkor lenne az elosztás, ha minden osztály *N*osztálylétszám/iskolalétszám* darab jegyet kapna. Ez azonban csak akkor lenne lehetséges, ha az összes így kiszámolt érték egész szám lenne. Ha törtértéket kapunk, akkor azt helyettesíteni kell valamelyik szomszédos egész számmal. Ekkor azt a jegyelosztást tekintjük igazságosnak, ahol az osztályonkénti jegyek száma az előző képlet által kiszámolt értéktől a lehető legkevesebbel tér el.

Készíts programot, amely beolvassa N és M értékét, majd pedig az M darab osztálylétszámot, s ezek alapján kiírja képernyőre, hogy az egyes osztályok az igazságos jegyelosztás szerint hány mozijegyet kapnak!

Példa:

```
Bemenet:                    a kiszámolt hányados:

M=4, N=20
1. osztálylétszám=20        20*20/80=5
2. osztálylétszám=25        20*25/80=6.25
3. osztálylétszám=18        20*18/80=4.5
4. osztálylétszám=17        20*17/80=4.25

Kimenet:

1. osztály: 5, 2. osztály: 6,3. osztály: 5, 4. osztály: 4
```

Kilencedik-tizedik osztályosok

1. feladat: OKTV (13 pont)

Seholsincs ország informatika OKTV-jén addig rendeznek új fordulókat, amíg vannak résztvevők, akiknek helyezése nem egyértelmű (holtversenyben vannak). Az újabb fordulókban csak ezeknek a résztvevőknek kötelező részt venniük, de új pontszámuk csak a holtverseny eldöntésére használható. (Az utolsó forduló után biztosan nincs holtverseny.) Részt vehetnek olyanok is a további fordulókban, akik helyezése már eldőlt. Ez utóbbi résztvevők ugyanúgy kapnak pontot, mint a többiek, de már nem változtathatnak helyezésükön. Ha úgy döntenek, hogy nem vesznek részt az újabb fordulóban, 0 pontot kapnak rá.

Az OKTV.BE állomány első sorában a versenyzők (1≤N≤1000) és a fordulók (1≤M≤10) száma van, a következő M sorban pedig az egyes fordulók eredménye. Minden eredménysor pontosan N nemnegatív számot tartalmaz, egy-egy szóközzel elválasztva.

Az OKTV.KI állományba és a képernyőre 1 sort kell írni, a versenyzők sorszámait helyezés szerinti sorrendben, egy-egy szóközzel elválasztva!

Példa:

```
OKTV0.BE:
5 3
7 3 3 9 7
2 7 7 0 1
0 3 2 0 8
```

```
OKTV0.KI:
4 1 5 2 3
```

2. feladat: Család (24 pont)

Családi kapcsolatokat úgy adunk meg, hogy mindenkihez felsoroljuk az anyja, illetve az apja nevét.

A CSALAD.BE állomány első sorában az ismert személyek száma ($1 \leq N \leq 100$) van. Ezután 3*N sorban jönnek az egyes emberek adatai: a neve, az apja neve és az anyja neve. Az utolsó sorban egyetlen ismert név szerepel, akinek valamilyen rokonaira kíváncsiak vagyunk. A nevek hossza legfeljebb 20 karakter.

Készíts programot, amely a képernyőre és a CSALAD.KI állományba az alábbi 4 sort írja:

A. A keresett személy testvérei száma és neve, egy-egy szóközzel elválasztva.

B. A keresett személy féltestvérei száma és neve, egy-egy szóközzel elválasztva. (A testvérek nem féltestvérek!)

C. A keresett személy férfiági felmenőinek listája (darabszám, majd apja, nagyapja, dédapja, ükapja, ... amíg ismert), egy-egy szóközzel elválasztva.

D. A keresett személy első unokatestvérei száma és neve (akikkel közös nagyszülője van az ismert személyek között), egy-egy szóközzel elválasztva.

Példa:

```
CSALAD0.BE:          CSALAD0.KI:
7                    1 Nagy András
Nagy Andrea          1 Nagy Erika
Nagy István          2 Nagy István Nagy Péter
Kiss Anna            1 Kovács Melinda
Nagy András
Nagy István
Kiss Anna
Nagy Erika
Nagy István
Szabó Éva
Nagy István
Nagy Péter
Kovács Eleonóra
Kiss Anna
Kiss Csaba
Takács Orsolya
Tóth Szilvia
Tóth Árpád
Takács Orsolya
Kovács Melinda
Kovács László
Tóth Szilvia
Nagy Andrea
```

3. feladat: Szójáték (18 pont)

A *Cserebere* logikai szójátékot egy szótár alapján játszhatjuk. A játék alapja: egy kiinduló szóból egy szósorozatot kell előállítani, amely csak a szótárban levő szavakból állhat, s a sorozat egymást követő szavai egy elemi átalakítással kaphatók az őket megelőző szóból. Kétféle szabály alkalmazható átalakításra:

A. szabály: egy betű kicserélése egy másikra;

B. szabály: egy betű kicserélése egy másikra vagy egy betű írása a szó végére.

A SZOTAR.BE állomány első sorában a szótár szavainak száma (1≤N≤1000), a további N sorban pedig egy-egy, a szótárban szereplő szó van. A szavak legfeljebb 20 karakterből állnak. A JATEK.BE állomány első sorában az alkalmazandó szabály betűjele (A vagy B), második és harmadik sorában pedig két különböző, a szótárban szereplő szó (P és Q) van.

Készíts programot, amely a képernyőre és a JATEK.KI állományba írja soronként az alábbiakat:

1. Az első sorba az IGEN szó kerüljön, ha a P szóból előállítható a Q szó a megadott szabály szerinti szósorozat előállítással, egyébként a NEM szót kell írni! Ha előállítható, akkor az IGEN szótól egy szóközzel elválasztva szerepeljen a legkisebb szabályalkalmazás szám, amivel a Q a P-ből előállítható!

2. A második sorba a P szóból a szabály szerint (nem feltétlenül egyetlen sorozatban) előállítható összes szót kell írni!

Példa:

```
SZOTAR0.BE:  JATEK0.BE:  JATEK0.KI:

7            A           IGEN 3
OKOS         OKOS        OKOS ÁKOS ÁKOM ÁLOM ALOM OKOD
ÁKOS         ÁLOM
ÁKOM
OKOD
ÁLOM
ALOM
HALOM
```

4. feladat: Kockák (20 pont)

Építőkockákból úgy lehet stabil tornyot építeni, hogy kisebb kockára nem lehet nagyobbat, illetve könnyebb kockára nem lehet nehezebbet tenni.

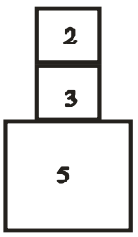
Készíts programot, amely N kocka alapján megadja a belőlük építhető legmagasabb tornyot!

A KOCKA.BE állomány első sorában a kockák száma (1≤N≤1000) van, a további N sorban pedig az egyes kockák oldalhossza és súlya (mindkettő 20 000-nél kisebb pozitív egész szám), egyetlen szóközzel elválasztva.

A KOCKA.KI állomány és a képernyő első sorába a legmagasabb torony M kockaszámát kell írni, a következő M sorba pedig az építés szerint alulról felfelé sorrendben a felhasznált kockák oldalhosszát és súlyát!

Példa:

```
KOCKA0.BE:     KOCKA0.KI:
5              3
10 3           20 5
20 5           10 3
15 6           10 2
15 1
10 2
```



Tizenegyedik-tizenharmadik osztályosok

1. feladat: Kockák (15 pont)

Építőkockákból úgy lehet stabil tornyot építeni, hogy kisebb kockára nem lehet nagyobbat, illetve könnyebb kockára nem lehet nehezebbet tenni.

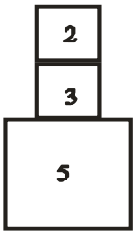
Készíts programot, amely N kocka alapján megadja a belőlük építhető legmagasabb tornyot!

A KOCKA.BE állomány első sorában a kockák száma (1≤N≤1000) van, a további N sorban pedig az egyes kockák oldalhossza és súlya (mindkettő 20000-nél kisebb pozitív egész szám), egyetlen szóközzel elválasztva.

A KOCKA.KI állomány és a képernyő első sorába a legmagasabb torony M kockaszámát kell írni, a következő M sorba pedig az építés szerint alulról felfelé sorrendben a felhasznált kockák oldalhosszát és súlyát!

Példa:

```
KOCKA0.BE:      KOCKA0.KI:
5               3
10 3            20 5
20 5            10 3
15 6            10 2
15 1
10 2
```



2. feladat: Barátságok (16 pont)

Egy pszichológiai vizsgálatban azt jegyezték fel, hogy egy N tanulóból álló csoportban ki kit tart a legszimpatikusabbnak.

A BARAT.BE állomány első sorában a tanulók száma (2≤N≤1000), a további N sorban a tanulók neve és a nekik legszimpatikusabb tanuló neve, egyetlen szóközzel elválasztva. A nevek szóközt nem tartalmazhatnak, hosszuk legfeljebb 20.

Készíts programot, amely a képernyőre és a BARAT.KI állományba írja (soronként) az alábbiakat:

A. Az első sorba a senkinek sem szimpatikus tanulók nevét, egy-egy szóközzel elválasztva.

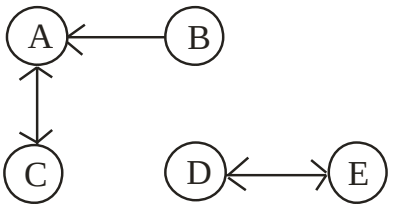
B. A második sorba azokat a párokat, amelyek tagjai egymásnak a legszimpatikusabbak, de senki másnak nem szimpatikus egyik sem. A párok tagjai közé kötőjelet (–), a párok közé pedig szóközt kell írni!

C. A harmadik sorba a legkedveltebb tanuló nevét (akit a legtöbben adtak meg legszimpatikusabbként), ha több van, akkor mindegyiket, egy-egy szóközzel elválasztva.

D. A negyedik sorba azt a legnagyobb számot, ahány csoportra lehet osztani a tanulókat úgy, hogy minden tanuló ugyanazon csoportba kerüljön, mint amelyikben a neki legszimpatikusabb van.

Példa:

```
BARAT0.BE:      BARAT0.KI:
5               B
A C             D-E
B A             A
C A             2
D E
E D
```



3. feladat: Terv (14 pont)

Egy nagyszabású építkezés azzal kezdődött, hogy kijelölték az építési területen az építhető épületek lehetséges helyeit. Minden lehetséges épület alaprajza téglalap alakú, megadható egy rögzített koordinátarendszerben az épület bal alsó sarkának (x, y) koordinátáival és az x tengellyel, illetve az y tengellyel párhuzamos oldalainak dx, illetve dy hosszával. Az építendő épületeket egymást nem takaró módon kell elhelyezni, azaz az origóból nézve egyik sem takarhatja bármely másik kiválasztott épület egyetlen pontját sem.

Írj programot, amely kiszámítja az egymást nem takaró módon elhelyezhető épületek legnagyobb számát!

A TERV.BE állomány első sora a lehetséges épület elhelyezések (1<N<5000) számát tartalmazza. A további N sor mindegyike négy pozitív egész számot tartalmaz: x y dx dy (0<x, y, dx, dy <20000) egy-egy szóközzel elválasztva, egy lehetséges épület elhelyezés adatait. Az első két szám az épület bal alsó sarkának x, illetve y koordinátája, a harmadik szám az x tengellyel, a negyedik pedig az y tengellyel párhuzamos oldal hossza.

A TERV.KI állomány és a képernyő első sora az egymást nem takaró módon elhelyezhető épületek legnagyobb K számát tartalmazza! A további K sorban kell megadni az elhelyezett épületek adatait, ugyanúgy, mint a bemeneti állományban!

Példa:

TERV0.BE:	TERV0.KI:
9	4
3 11 2 3	1 8 2 4
1 8 2 4	6 10 2 2
4 2 1 1	4 2 1 1
6 10 2 2	11 1 3 2
6 6 1 3	
6 6 1 7	
7 9 2 3	
11 1 3 2	
7 4 2 1	

4. feladat: Kamion (15 pont)

Egy vállalat az ország különböző városaiban levő üzemeiben alkatrészeket termel. A heti termelést a hét végén kamionokkal szállítja a központi raktárába. A kamionforgalom korlátozása miatt minden városból pontosan egy másik városba (egy irányban) mehetnek a kamionok közvetlenül. Ezért a vállalat úgy tervezi a szállításokat, hogy minden olyan városból, amelybe más városból nem lehet eljutni, egy-egy kamiont indít, a többi városból viszont egyet sem. A korlátozások miatt így minden kamion útja a központi raktárig egyértelműen meghatározott.

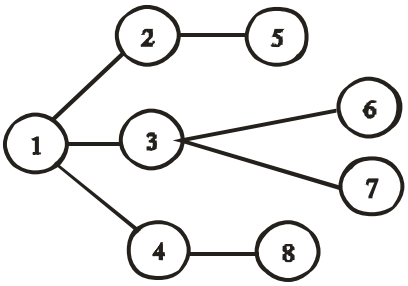
Minden kamion, amely útja során áthalad egy városon, az ott termelt alkatrészekből bármennyit felvehet, feltéve, hogy nincs tele. Ismerve a városokban termelt alkatrészek számát, ki kell számítani azt a legkisebb kamion kapacitást, amellyel a szállítás megoldható, ha minden kamion azonos kapacitású!

A KAMION.BE állomány első sorában a városok száma (1<N≤200) van A központi raktár az 1. városban van, és onnan nem kell szállítani. Az állomány következő N-1 sorának mindegyike két egész számot tartalmaz, egy szóközzel elválasztva. Az állomány I-edik sorában az első szám azt a várost adja meg, ahova az I-edik városból mehet kamion. A második szám pedig az I-edik városban termelt alkatrészek száma. (Az 1. városból kivezető út nincs megadva.)

A KAMION.KI állományba és a képernyőre azt a legkisebb kamion kapacitást (egész szám) kell írni, amekkora kapacitású kamionokkal az összes alkatrész elszállítható!

Példa:

```
KAMION0.BE      KAMION0.KI
8               12
1 2
1 3
1 4
2 5
3 6
3 7
4 8
```



5. feladat: Akadálypálya (15 pont)

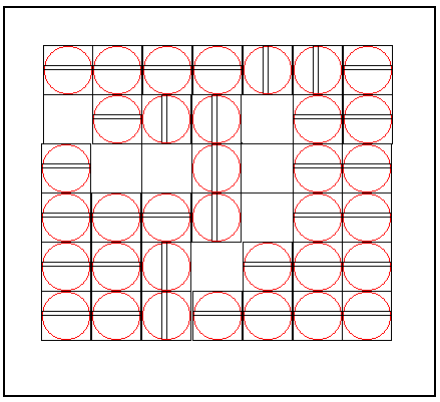
Egy jármű négyzetrácsos elrendezésű pályaelemekből álló, M sorból, N oszlopból álló pályán mozoghat. Minden pályaelem vagy üres, vagy a közepén áthaladó sínt tartalmaz, amelyen a jármű haladhat. Egy pályaelem négy szomszédja a négyzetrácsos elrendezésben a tőle balra, jobbra, lefelé vagy fölfelé lévő pályaelem. A jármű egy lépésben a következő három lehetséges mozgást végezheti:

1. 90 fokkal elfordítja azt a pályaelemet, amelyen éppen áll.

2. Átmegy egy szomszédos pályaelemre, feltéve, hogy azon a sín olyan irányban áll, hogy az csatlakozik az aktuális pályaelemen lévő sínhez.

3. 90 fokkal elfordít egy szomszédos pályaelemet.

Írj programot, amely kiszámítja azt a legkevesebb lépésszámot, amely megtételével a jármű a pálya bal felső (1,1) pontjából eljuthat a jobb alsó (M,N) pontjába.



A PALYA.BE állomány első sorában a pálya méretét megadó M N számpár van egy szóközzel elválasztva (1≤M, N≤100). A következő M sor mindegyike N számot tartalmaz egy-egy szóközzel elválasztva:

0: az adott pályaelem nem tartalmaz sínt (üres),

1: a pályaelemen a sín vízszintes irányban áll,

2: a pályaelemen a sín függőleges irányban áll.

A képernyőre és a PALYA.KI állomány első és egyetlen sorába azt a legkisebb lépésszámot kell írni, amely megtételével a jármű a pálya bal felső pontjából eljuthat a jobb alsó pontjába! Ha a jármű nem tud eljutni, akkor a -1 értéket kell kiírni!

Példa:

```
PALYA0.BE:          PALYA0.KI:
6 7                 17
1 1 1 1 2 2 1
0 1 2 2 0 1 1
1 0 0 2 0 1 1
1 1 1 2 0 1 1
1 1 2 0 1 1 1
1 1 2 1 1 1 1
```

1999. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Kiszámolós (20 pont)

Egy kiszámolós játékban N gyerek körbe áll az ábrának megfelelően:

A kiszámolás az elsőnél kezdődik, majd minden K-adikat kell kihagyni úgy, hogy végül csak egyetlen egy gyerek maradjon. Először tehát a K. marad ki, majd a 2*K., ... Ha az utolsóhoz értünk, a kör tovább folytatódik.

Példa:

N=13, K=6
A kimaradók sorban: 6, 12, 5, 13, 8, 3, 1, 11, 2, 7, 4, 10
Végül megmaradt: 9

Készíts programot, amely beolvassa a gyerekek számát (1<N≤100) és hogy minden hányadikat kell kihagyni (K≥1), majd kiírja képernyőre a kiszámolós játékban kiesőket, majd pedig a végén megmaradt gyerek sorszámát!

2. feladat: Telefonáló robot (25 pont)

Egy telefonkészüléken az ábrán látható elrendezésben vannak a nyomógombok. Egy robotkart kell irányítanunk, amely egy adott telefonszám számjegyeit nyomja le a készüléken.

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**
*****	**0**	**#**

A robotot az alábbi utasításokkal vezérelhetjük:

- E x x egységnyit mozdítja a robotkart északi irányba (negatív x esetén déli irányba)
- K x x egységnyit mozdítja a robotkart keleti irányba (negatív x esetén nyugati irányba)
- N lenyomja a nyomógombot, amelyik felett a robotkar áll.

A robotkar kezdetben a * jel fölött van, a telefonálás végén a # jel fölött kell lennie. Készíts programot, amely beolvas egy legfeljebb 7 jegyű telefonszámot, majd a minimális utasításszámmal végrehajtatja a robottal a telefonálást. Eredményként a robot számára szóló utasításokat kell kiírni a képernyőre.

3. feladat: Ismételgető (30 pont)

Készíts programot, amely beolvas egy legfeljebb 150 karakterből (az angol ABC kisbetűiből) álló szöveget, majd ABC-sorrendben kiírja mindazon legalább 2 karakterből álló részeit, amelyek egynél többször fordulnak elő!

Példa:

```
Bemenet: mississippi
Kimenet: is, iss, issi, si, ss, ssi
```

Kilencedik-tizedik osztályosok

1. feladat: Hálózati felügyelőprogram (23 pont)

Egy számítógép-hálózaton a szerver nyomon követi a felhasználók be- és kijelentkezését, melynek alapján naponta többféle jellemzőt kiszámíthatunk. Minden felhasználó a munkája végén köteles

kijelentkezni, valamint egyszerre csak egyetlen gépen jelentkezhet be. Ha az első adata egy kijelentkezés, akkor azt úgy kell érteni, hogy még az előző napon jelentkezett be, s ha nem jelentkezett ki, az azt jelenti, hogy még a következő napon is folytatja a munkáját.

Készíts programot (SZERVER.PAS vagy SZERVER.C), amely egy napi be- és kijelentkezési adatok alapján megadja a rendszer egy napi statisztikáját!

A SZERVER.BE állomány minden sorában egy-egy be- vagy kijelentkezés adatai vannak. A sor első két karaktere a BE vagy a KI szó, majd ezt követi egy-egy szóközzel elválasztva a felhasználó azonosítója (legfeljebb 6 karakter), a művelet óra és perc adata. Az állomány legfeljebb 3200 sort tartalmazhat, s legfeljebb 1500 felhasználói azonosítót adtak ki.

A SZERVER.KI állomány első sorába azon legkevesebb intervallumok számát kell írni, amely intervallum minden percében használta a rendszert legalább 1 felhasználó, s tőle egy-egy szóközzel elválasztva az intervallumok kezdetét és végét!

A második sorba azon intervallumok számát kell írni, amelyekben a rendszert a legtöbb felhasználó használta, s tőle egy-egy szóközzel elválasztva az intervallumok kezdetét és végét!

A harmadik sorba a legtöbb összesített rendszeridőt használó felhasználó azonosítóját kell írni! Ha több ilyen van, akkor mindet meg kell adni, egy-egy szóközzel elválasztva!

A negyedik sorba a tiltott tevékenységet végző felhasználók adatait (művelet, azonosító és idő) kell kiírni (bejelentkezik úgy, hogy előtte nem jelentkezett ki, illetve kétszer próbál kijelentkezni), egy-egy szóközzel elválasztva! A hibás műveleteket a szerver visszautasítja, ezek a többi részfeladat megoldásában nem játszanak szerepet.

Példa:

```
SZERVER.BE          SZERVER.KI

BE ALFA 3 15        2 0 0 5 30 6 30 11 45
KI BETA 4 50        2 3 15 4 50 6 35 6 55
KI ALFA 5 30        ALFA
BE GAMMA 6 30       KI GAMMA 7 55
BE ALFA 6 35
KI GAMMA 6 55
KI GAMMA 7 55
KI ALFA 11 45
```

2. feladat: Katonák (28 pont)

Terepen NxN-es négyzetrácsos elrendezésben N katona helyezkedik el, különböző helyeken. A parancsnok olyan elrendezésbe kívánja parancsolni a katonákat, hogy minden sorban és minden oszlopban legyen katona. Egy időpontban egyszerre csak egy katona léphet a négy szomszédos mező valamelyikére, és minden egyes lépést egy időegység alatt hajt végre a katona. Az a cél, hogy a legrövidebb időn belül elérjenek egy kívánt elhelyezkedést. A parancsnok négy adatot: `x y i h`, tartalmazó parancsot adhat ki a katonáknak annak érdekében, hogy a kívánt elrendezést elérje, ahol

- `x y` : a mozgatandó katona koordinátái
- `i` : a mozgás iránya, ami lehet
  - L : lefelé, y-irányban csökkenően
  - F : felfelé, y-irányban növekvően
  - B : balra, x-irányban csökkenően
  - J : jobbra, x-irányban növekvően
- `h` : a lépések száma a megadott irányban

Tehát minden egyes `x y i h` parancs végrehajtása `h` ideig tart.

Csak olyan parancsot lehet kiadni, amely végrehajtása során nem kerül két katona ugyanabba a pozícióba, és a mozgó katona nem is léphet át másik katonán.

Írj programot (KATONAK.PAS vagy KATONAK.C), amely megoldja az alábbi 3 részfeladatot:

A. A legkisebb idő, amely a feladat megoldásához kell.

B. A katonák egy lehetséges elhelyezkedése, ami a legrövidebb idő alatt elérhető.

C. A parancsok sorozata, amely a megoldást adja.

A KATONAK.BE állomány első sorában a katonák ($2 \le N \le 200$) száma van. A következő N sor a katonák kezdeti elhelyezkedését adja meg, egy sorban egy katona x,y koordinátái vannak egy szóközzel elválasztva.

A KATONAK.KI állomány első sora az A részfeladat megoldását, azaz a legrövidebb időt tartalmazza! Az állomány második sorába a B. részfeladat megoldását adó N számot kell írni egy-egy szóközzel elválasztva! Az i-edik szám azon katona y-koordinátája legyen, amelynek x koordinátája a kívánt elhelyezkedésben! A harmadik sorba a parancsok P számát kell írni, amivel elérhető, hogy minden sorban és minden oszlopban legyen katona! A negyedik sortól P sorban kell megadni a parancsokat!

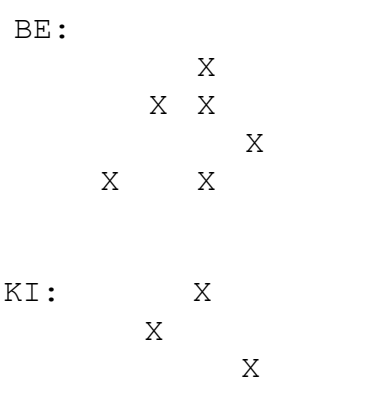
Példa:

KATONAK.BE

```
6
1 2
2 4
3 4
3 5
4 3
3 2
```

KATONAK.KI

```
8
1 5 6 4 2 3
6
4 3 J 2
3 2 J 2
3 4 J 1
1 2 L 1
3 5 F 1
2 4 F 1
```



3. feladat: Fogadás (24 pont)

Byteland ország követsége fogadást rendezett. A fogadásra N számú ország nagykövetét hívták meg. A vendégek különböző időpontokban érkeztek, és minden vendég azonos ideig volt jelen a fogadáson. A házigazda feljegyezte, hogy ki-kivel találkozott a fogadás során. Az előbb érkezett A vendég találkozott a később érkezett B vendéggel, ha A később távozott, mint B érkezett. Azt is tudjuk, hogy az első vendég érkezésétől az utolsó távozásáig minden időpontban jelen volt legalább egy vendég.

A feladat annak kiderítése, hogy a vendégek milyen sorrendben érkezhettek a fogadásra.

Írj programot (FOGAD.PAS vagy FOGAD.C), amely kiszámítja a vendégek egy lehetséges érkezési sorrendjét (nem az érkezési időpontokat)!

A FOGAD.BE állomány első sora a vendégek ($2 < N \le 200$) számát tartalmazza. A második sorban a fogadás alatt találkozott vendégpárok száma áll ($1 \le M \le 10\,000$). A következő M sor mindegyike egy X Y számpárt ($1 \le X,Y \le N$) tartalmaz egy szóközzel elválasztva, ami azt jelenti, hogy az X és az Y vendég találkozott a fogadás során.

A FOGAD.KI állomány első és egyetlen sorába N számot kell írni egy-egy szóközzel elválasztva, ami a vendégek egy lehetséges érkezési sorrendje!

Példa:

```
FOGAD.BE        FOGAD.KI
4               1 2 4 3     (3 4 2 1 is jó megoldás)
3
1 2
2 4
3 4
```

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Elfogó (16 pont)

A városi rendőrség egy veszélyes bűnöző elfogását tervezi, aki gépkocsival folyamatosan közlekedik a város utcáin. A rendőrségnek korlátozottak a lehetőségei, nem tud például minden kereszteződésbe rendőrt állítani az elfogás érdekében. Ravasz őrmesternek az alábbi kitűnő ötlete támadt. Egy kijelölt K kereszteződésből indulva bejárja a város utcáit és úgy egyirányúsítja azokat, hogy a bűnöző előbb-utóbb úgyis eljut a K kereszteződésbe, ahol egy másik rendőr várakozik, aki elfogja a bűnözőt. A kapitánynak nagyon tetszik az ötlet, de kiköti, hogy Ravasz őrmesternek is be kell tartania a közlekedési szabályokat:

A már egyirányúsított utcában az őrmester is csak a jelzett irányban közlekedhet. Ha az A kereszteződésből a B-be vezető utcát akarja egyirányúsítani, azt csak úgy teheti, hogy elmegy az A-ba, ott elhelyez egy behajtani tilos táblát B irányában, ezután elmegy az utcában a B kereszteződésig és ott elhelyezi az A-irányába mutató egyirányú utca táblát. Ugyanazon utcában többször is járhat, de csak a már beállított irányban.

Így a megoldás egyértelműen leírható az őrmester útja során érintett kereszteződések sorozatával, mivel ha az útja során egy olyan A kereszteződésbe ér, amelyből a B irányában halad tovább, akkor ha ez az utca még nem volt egyirányúsítva, akkor egyirányúsítja, egyébként csak halad tovább az utcában. A város úthálózata összefüggő, azaz minden kereszteződésből el lehet jutni bármely másik kereszteződésbe.

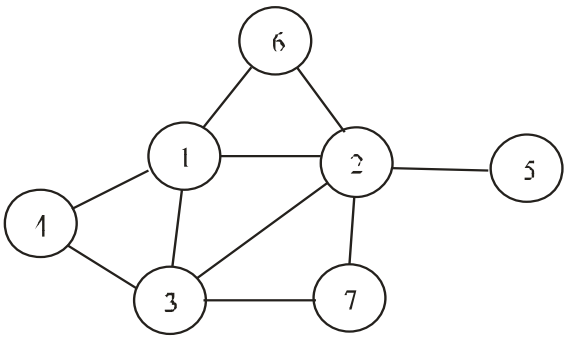
Írj programot (ELFOGO.PAS vagy ELFOGO.C), amely kiszámít egy olyan bejárási sorozatot, amelyet bejárva és elvégezve az egyirányúsítást, a bűnöző előbb-utóbb feltűnik abban a kereszteződésben, ahonnan az őrmester indult! Minden utcában mindkét irányban lehet közlekedni, de az utcán megfordulni tilos. Kereszteződésben azonban meg lehet fordulni. Az egyirányúsítás következtében csak az a kereszteződés lehet olyan, hogy nem indul belőle egyetlen egyirányú út sem, amelyikből az őrmester indult.

Az ELFOGO.BE állomány első sorában a kereszteződések ($1 \le N \le 200$) száma, a második sorában az utcák ($1 \le M \le 10\,000$) száma van. A következő M sor mindegyike egy A B számpárt ($1 \le A, B \le 200$) tartalmaz egy szóközzel elválasztva, ami azt jelenti, hogy az A kereszteződésből megy egy (kétirányú) utca a B kereszteződésig. Két kereszteződés között legfeljebb egy utca lehet.

Az ELFOGO.KI állomány első sorába az őrmester bejáró útját megadó kereszteződések L számát, a második sorba pedig a bejárt L kereszteződés sorszámát kell írni, egy-egy szóközzel elválasztva.

Példa:

```
ELFOGO.BE       ELFOGO.KI
7               21
10              1 6 2 7 3 4 3 7 2 5 2 3 2 6 1 4 1 3 1 2 1
1 2
1 3
1 4
2 5
2 6
3 7
1 6
2 3
3 4
2 7
```



2. feladat: Ültetés (24 pont)

Az iskola színháztermében N számú ülőhely van.
A következő előadásra M tanuló kérhet jegyet, és mindegyik meghívott tanuló két hely sorszámát megadhatja, mint az általa előnyben részesítettet.

Írj programot (ULTET.PAS vagy ULTET.C), amely kiszámítja, hogy legjobb esetben hány tanuló kaphat olyan jegyet, amely az igénylésének megfelel! A program azt is megadja, hogy mely tanulók kapják az igénylésüknek megfelelő helyeket. Kiszámítandó továbbá, hogy lehet-e az igényeket úgy kielégíteni, hogy a kiosztott helyek összefüggő tartományt alkossanak!

Az ULTET.BE állomány első sorában az ülőhelyek száma ($1 \le N \le 200$), második sorában a tanulók száma ($1 \le M \le 250$) van. A következő M sor mindegyike két különböző ülőhely sorszámot (A B) tartalmaz egy szóközzel elválasztva ($1 \le A, B \le N$). Az állomány i+2-edik sora az i-edik tanuló kívánságát tartalmazza.

Az ULTET.KI állomány első sorába azon tanulók T számát kell írni, ahányan a legjobb esetben megkaphatják a két kívánságuknak megfelelő ülőhely valamelyikét! A második sorba egy legjobb kiosztás szerinti ültetést kell írni, tehát T számpárt, amelynek első tagja egy tanuló sorszáma, második tagja pedig azt az általa kívánt ülőhelysorszámot tartalmazza, amit a tanuló kap a legjobb kiosztás szerint! A második sorban a számokat egy-egy szóköz válassza el! A harmadik sorba az IGEN szót kell írni, ha van olyan legjobb kiosztás, amely szerint nincs üresen maradt szék a foglalt helyek között, egyébként pedig a NEM szót!

Példa:

```
ULTET.BE        ULTET.KI
10              8
10              1 1 2 2 7 3 3 4 4 5 5 6 6 7 8 10
1 4             NEM
2 4
4 6
6 5
6 7
7 5
3 5
8 10
1 4
4 7
```

3. feladat: Autómentés (35 pont)

Az autómentő szolgálat számítógépes kapcsolatban van az autópálya üzemeltetőjével, aki minden, a pályára belépő autó adatát rögzíti egy adatbázisban. Ezeket az adatokat felhasználva tervezi meg egy autómentő útját a baleset helyszínéig.

Az autópálya egy 2-5 sávos, egyenes és egyirányú forgalmat lebonyolító, elágazás nélküli útszakasz. A 0 sáv a leállósáv, ebben azonban nem közlekedhet semmilyen autó. Az autópályán egy autó helyzetét az autópálya kezdetétől számolva méterekben, illetve a külső sávtól számítva egy sávszámmal tudjuk megadni. Például a következő autó helyzetét a (4,2) koordináta adja meg:

Minden autó az alábbi szabályok betartásával közlekedik:

- Minden időpontban, minden pozícióban legfeljebb egy autó tartózkodhat.
- A leállósávban (0-s sáv) nem közlekedhet autó.
- Minden autó a számára adott (belépéskor rögzített) sebességgel akar közlekedni. Ez azt jelenti, hogy ha az A autó előtt közvetlenül nincs autó, akkor a sávjában előre halad. Tehát ha a sebessége v és a t időpontban a pálya (x, y) pontjában tartózkodik, akkor a t+1 időpontban az (x1, y) pontba jut, ahol x1=x+v ha a t+1 időpontban nincs előtte az y sávban autó az x+1, …, x+v pozíciókban, egyébként pedig x1=Minimum(x+v, u-1) ha u a legkisebb, x-nél nagyobb olyan pozíció, ahol van autó a t+1 időpontban.
- Ha A utolérte B-t, azaz A pozíciója (x, y) és B pozíciója (x+1, y), akkor A előzni próbálja B-t ha sebessége nagyobb, mint B sebessége. Először balra, ha nem megy akkor jobbra próbál előzni. Autó csak akkor válthat sávot, ha nincs mellette autó abban a sávban, amibe lépni akar. Balra előzés azt jelenti, hogy átlép az y+1 sávba az x+1 pozícióba, jobbra előzésnél pedig az y-1 sávra az x+1 pozícióba. Szomszédos sávba csak akkor léphet át az autó, ha nem zavarja az ott közlekedő autók forgalmát, tehát ha abban a sávban az x+1 pozíción nem halad át autó a t-t+1 időben. (Egy autó áthalad a t-t+1 időben az x pozíción, ha a t időpontban az x1, a t+1 időpontban pedig az x2 pozíción van és $x1<x\leq x2$.) Előzés esetén a balra előzőnek van elsőbbsége, tehát ha két autó ugyanazon pozícióba kerülne előzés következtében, akkor az előzést csak a balra előző hajthatja végre, a másik nem. Ha az autó nem tud előzni, akkor természetesen követi az előtte haladót.

Ha baleset következik be a ***t*** időpontban, akkor az autómentő azonnal indul az autópálya bejáratától valamelyik (általa választott) sávban. (Tehát a t+1 időpontban lép be a pálya 1-es pozíciójába.) A baleset helyszíne a 0-s sáv egy pozíciója. A baleset időpontjában lezárják az autópályát, tehát ezt követően nem léphet autó a pályára.

Az autómentő hasonló szabályok szerint közlekedik, mint a többi autó, de rá az alábbiak vonatkoznak:

- Semmilyen formában nem befolyásolhatja a többi autó haladását.
- Minden időpontban eldöntheti, hogy
  - vagy előre halad a sávjában tetszőleges, de a számára rögzítettnél nem nagyobb sebességgel, figyelembe véve az előtte haladó autót;
  - vagy sávot változtat akár balra akár jobbra.

  Sávváltáskor természetesen nem zavarhatja a többi autó haladását.

- Ha az 1-es sávon lépésével áthaladna vagy elérné a baleset helyét jelentő x pozíción, akkor e helyett leléphet a 0-s sáv ***x*** pozíciójába (a baleset helyére).

A programodnak a következőkre kell választ adnia:

A.

Kérdések:

- Hány autó tartózkodik a baleset időpontjában az autópályának a bejáratától a baleset helyszínéig terjedő szakaszán?
- Hol tartózkodnak az autók a baleset időpontjában?

B. A baleset időpontjában riadó jelzést adunk le, ezzel az autóvezetőket arra kötelezve, hogy azonnal álljanak meg. Ekkor indulunk autómentőnkkel a baleset helyszínére. A baleset helye mindig a 0-s sáv egy pozíciója, amire a 1-es sávból lép a mentő, és ez a lépés is egy időegységet igényel.

Kérdések:

- Minimálisan mennyi idő alatt juthat el az autómentő a baleset színhelyére?
- Hogyan juthat-e az autómentő a baleset helyszínére?

C. A baleset időpontjában valamennyi autó jelzést kap, hogy ezt követően tilos sávot változtatniuk, továbbá állandó sebességgel kell haladniuk, ami eggyel kisebb, mint az autómentő sebessége.

Kérdés:

- Minimálisan mennyi idő alatt juthat el az autómentő a baleset színhelyére?

D. Nem korlátozzuk az autók forgalmát, a továbbhaladó forgalomban próbálunk meg eljutni a baleset helyszínéhez.

Kérdés:

- Minimálisan mennyi idő alatt juthat el az autómentő a baleset színhelyére?

Az AUTO.BE állomány első sorában a sávok száma ($1<K\leq5$) van A második sorban az autómentő maximális sebessége áll. A harmadikban a baleset időpontja és pozíciója van egy szóközzel elválasztva. A baleset időpontja kisebb, mint 1000.

A további sorok mindegyike egy autó adatait, három pozitív egész számot tartalmaz: T S V, ahol T a pályára lépés időpontja, S az a sáv, amelyiken belépett az autó, V pedig a megengedett legnagyobb sebessége. Az utolsó sor 3 db 0-t tartalmaz. Minden sebesség érték nagyobb, mint 0 és kisebb, mint 100. Az adatok az állományban a belépési idő szerint nemcsökkenő sorrendben vannak.

A állomány maximum 4000 soros lehet, az autópálya pedig 4000 m hosszú lehet.

Az AUTO.KI állományban az A és B részfeladathoz 2, a többihez 1 sor tartozik. Ha a program valamelyik részfeladat kérdésére nem tud válaszolni, akkor a részfeladathoz üresen sort kell írni!

Az első sorba egy egész számot, a baleset időpontjában az autópályának a bejáratától a baleset helyszínéig terjedő szakaszán tartózkodó autók számát kell írni! A második sorba ezen autók koordinátáit kell kiírni egy-egy szóközzel elválasztva! Ha nincs autó a pálya ezen szakaszán, akkor üres sort kell írni!

A harmadik sorba azt a legkisebb időt (baleset helyszínére érés időpontja – a baleset időpontja) kell írni, amely ahhoz szükséges, hogy az autómentő eljusson a baleset helyére, feltéve, hogy minden autó áll a baleset bekövetkezése után! Ha nem lehet eljutni, akkor a –1 értéket kell kiírni! A negyedik sorba azt a koordinátasorozatot kell kiírni, amelyeken keresztül az autómentő eljut a baleset helyére! Az utolsó koordináta-pár a baleset helyszínének a koordinátái, tehát Bx 0, ha a baleset a Bx helyen történt. Ha nem eljutni, akkor üres sort kell írni!

Az ötödik sorba egyetlen egész számot, azt a legkisebb időt kell írni, amely ahhoz szükséges, hogy az autómentő eljusson a baleset helyére, feltéve, hogy az autók a baleset bekövetkezése után nem válthatnak sávot és állandó (a mentő sebessége-1) sebességgel közlekednek! Ha nem lehet eljutni, akkor a –1 értéket kell kiírni!

A hatodik sorba egyetlen egész számot, azt a legkisebb időt kell írni, amely ahhoz szükséges, hogy az autómentő eljusson a baleset helyére, ha együtt kell haladnia a forgalommal! Ha nem lehet eljutni, akkor a –1 értéket kell kiírni!

Példa:

```
AUTO.BE          AUTO.KI

4                8
4                2 1 3 1 5 1 4 2 2 3 4 3 5 3 5 4
5 13             7
1 3 1            1 2 3 2 4 1 5 2 6 1 9 1 13 0
1 1 1            5
2 3 1            6
3 1 1
3 4 2
4 2 3
4 3 1
4 1 3
0 0 0
```

A verseny végeredménye:

I. kategória

1. Ritzinger Péter	Apor Vilmos Iskolaközpont, Győr
Dézsi Richárd	Baksay Sándor Református Gimnázium, Kunszentmiklós
3. Simon Balázs	Révai Miklós Gimnázium, Győr
Kovács Márton	Fényi Gyula Jezsuita Gimnázium, Miskolc
Nagy Ákos	Petőfi Sándor Általános Iskola, Debrecen
6. Sztupák Szilárd	Hermann Ottó Gimnázium, Miskolc
Bárkai János	Gárdonyi Géza Általános Iskola, Budapest
8. Préda Máté	Óbudai Gimnázium, Budapest
Siklós Jácint	Kinizsi Lakótelepi Általános Iskola, Kaposvár
10. Szatmári Zsolt	Lehel Vezér Gimnázium, Jászberény
Rátky Gábor	Eötvös József Gimnázium, Budapest
Bujtás Balázs	Lébényi Általános Iskola, Lébény
Bálint Márton	Fazekas Mihály Gimnázium, Budapest

II. kategória

1. Erdélyi Róbert	Bibó István Gimnázium, Kiskunhalas
2. Rokob András	Földes Ferenc Gimnázium, Miskolc
3. Novák Ádám	Neumann János Szakközépiskola, Eger
4. Pallos Péter	Fazekas Mihály Gimnázium, Budapest
5. Csirmaz Előd	Fazekas Mihály Gimnázium, Budapest
6. Pollák Gergely	Ságvári Endre Gimnázium, Szeged

7. Turi Péter	Révai Miklós Gimnázium, Győr
Novák Zoltán	Zrínyi Miklós Gimnázium, Zalaegerszeg
9. Gyebnár Gábor	Ságvári Endre Gimnázium, Szeged
10. Pszota Zsolt	Boronkai György Szakközépiskola, Vác
III. kategória	
1. Rácz Balázs	Veres Péter Gimnázium, Budapest
2. Ágó Péter	Petrik Lajos Szakközépiskola, Budapest
Gunda Lénárd	KLTE Gyakorló Gimnázium, Debrecen
Varga Kornél	Földes Ferenc Gimnázium, Miskolc
5. Darabos Dániel	Bárdos László Gimnázium, Tatabánya
6. Dezső Balázs	Teleki Blanka Gimnázium, Székesfehérvár
7. Fazekas Ferenc	Mechwart András Szakközépiskola, Debrecen
8. Förhécz András	Teleki Blanka Gimnázium, Székesfehérvár
9. Sáfár Szilveszter	Ságvári Endre Gimnázium, Szeged
10. Merksz Andor	Bencés Gimnázium, Pannonhalma

II.

Megoldások, értékelések

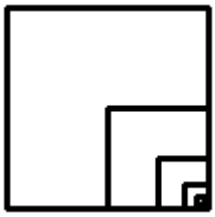
Nemes Tihamér
Nemzetközi Informatikai Tanulmányi Verseny

1995. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Logo (36 pont)

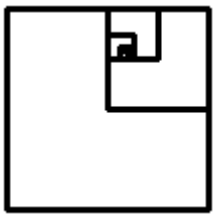
A. A jobb alsó sarokban befelé haladó, feleződő rekurzív négyzetek 12 pont



Részpontszámok:

négyzetek	2 pont
egymás belsejében	2 pont
a méreteik feleződnek	2 pont

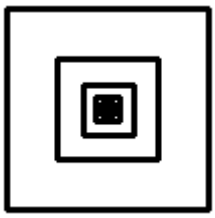
B. Jobbra körben befelé haladó, feleződő rekurzív négyzetek 12 pont



Részpontszámok:

négyzetek	2 pont
egymás belsejében	2 pont
a méreteik feleződnek	2 pont

C. Középen befelé haladó, feleződő rekurzív négyzetek 12 pont



Részpontszámok:

négyzetek	2 pont
egymás belsejében	2 pont
a méreteik feleződnek	2 pont

2. feladat: Csere (20 pont)

A. igen 3 pont

B. nem 3 pont

C. igen 3 pont

D. nem 3 pont

E. nem állíthatók elő az olyan sorrendű sorozatok, amelyekben az eredetihez képest bármelyik elem több, mint egy hellyel mozdulna el előre. 8 pont

3. feladat: Síknegyed (20 pont)

A. Az első síknegyedben levőkre a IV. síknegyed választ is kiírja 5 pont

B. A negyedik síknegyedben levő pontokra semmit sem ír ki (mert a IV. síknegyed feltétele rossz – Y<0 kellene) 5 pont

C. Nem mond semmit, ha az X tengelyen van a pont 5 pont

D. Nem mond semmit, ha az Y tengelyen van a pont 5 pont

4. feladat: Útkereszteződés (24 pont)

A. Keresztező főút balról – főútvonal szabály 3 pont

Egyszerre 4 irányból jön autó 3 pont

B. Keresztező főút balról – főútvonal szabály 3 pont

Keresztező főút jobbról – főútvonal szabály 3 pont

Autó jön egyenrangú úton jobbról – jobbkézszabály 3 pont

Van két szemből jövő, akik balra akarnak fordulni 3 pont

C. Keresztező főút balról – főútvonal szabály 3 pont

Jobbról felesleges várni autót 3 pont

Kilencedik-tizedik osztályosok

1. feladat: Kitérő (12 pont)

A. igen 1 pont

BE, BE, BE, ÁT (lehet helyette BE, KI), KI, KI, KI 2 pont

B. nem 1 pont

A 4 kijutásához BE, BE, BE kell, de ekkor a kitérő végén a 3 található, s már csak a KI művelet használható. 2 pont

C. igen 1 pont

ÁT (lehet helyette BE, KI), BE, BE, ÁT, KI, KI 2 pont

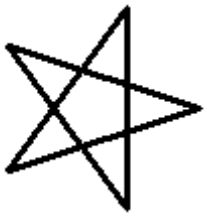
D. nem 1 pont

Az 1 kijutása után a 4 kijutásához a 2-t és a 3-at a kitérőbe kell irányítanunk, de a kitérőből először a 3 jöhetne ki. 2 pont

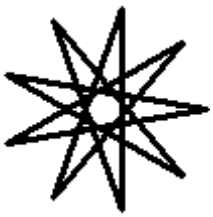
2. feladat: Logo (16 pont)

Az eredmény a legkisebb olyan szám lesz, amivel a fordulat szögét szorozva 360 fok többszörösét kapjuk.

A. Ötös 4 pont

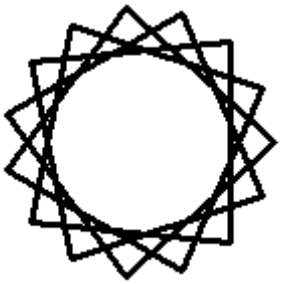


B. Kilences 4 pont



C. Tizenkettes 4 pont

D. Tizenötös 4 pont



3. feladat: Útkereszteződés (10 pont)

A. Keresztező főút balról – főútvonal szabály 1 pont
Egyszerre 4 irányból jön autó 3 pont
B. Keresztező főút balról – főútvonal szabály 1 pont
Keresztező főút jobbról – főútvonal szabály 1 pont
Autó jön egyenrangú úton jobbról – jobbkézszabály 1 pont
Van két szemből jövő, akik balra akarnak fordulni 1 pont
C. Keresztező főút balról – főútvonal szabály 1 pont
Jobbról felesleges várni autót 1 pont

4. feladat: Transzformáció (12 pont)

A. FORGAT((1,1),(2,2)) 3 pont
XTÜKÖR((2,1),(3,2)) 3 pont
B. YTÜKÖR((1,1),(3,2)) 3 pont
FORGAT((2,2),(3,3)) 3 pont

5. feladat: Szavak (20 pont)

A. "MACSKA 2 pont
a mondat második szava 3 pont
B. "ÉGIGÉRŐ 2 pont
az első szó az első betűje nélkül 3 pont
C. ~O 2 pont

az első szó utolsó betűje 3 pont

D. [LESZ] 2 pont

a mondat az első és az utolsó szava nélkül 3 pont

6. feladat: Mit csinál? (14 pont)

A. A program az A, egész számokat tartalmazó vektor 1 pont

B. elemei közül megkeresi 2 pont

C. a legkisebb 2 pont

D. pozitív számot, 1 pont

E. méghozzá a legkisebb indexű előfordulását 2 pont

F. Az l változó azt jelzi, hogy talált-e a program megoldást. 2 pont

G. Az m változó a talált érték, 2 pont

H. az n pedig annak helye (indexe A-ban). 2 pont

7. feladat: Levelezés (16 pont)

A. Pisti nem érti meg a *beszéljünk* üzenetet. 2 pont

B. Józsi nem érti meg a *jöhet* üzenetet. 2 pont

C. Józsi nem érti meg a *nemértem* üzenetet. 1 pont

D. Pisti nem érti meg az *ismétlem* üzenetet. 2 pont

E. Pisti nem érti meg a *vége* üzenetet. 1 pont

F. Több, mint 5 levél van útközben. 2 pont

G. Pisti akkor küldi a *nemértem* üzenetet, amikor Józsi már elküldte a *vége* üzenetet. 3 pont

H. A *nemértem* üzenet sorszáma nem olvasható vagy hibás. 3 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rendzőpályaudvar (12 pont)

A. igen 1 pont

BE, BE, BE, ÁT (lehet helyette BE, KI), KI, KI, KI 2 pont

B. nem 1 pont

A 4 kijutásához BE, BE, BE kell, de ekkor a kitérő végén a 3 található, s már csak a KI művelet használható. 2 pont

C. igen 1 pont

ÁT (lehet helyette BE, KI), BE, BE, ÁT, KI, KI 2 pont

D. nem 1 pont

Az 1 kijutása után a 4 kijutásához a 2-t és a 3-at a kitérőbe kell irányítanunk, de a kitérőből először a 3 jöhetne ki. 2 pont

2. feladat: Kártya (12 pont)

A. A legrégebben használt lapját cseréli ki a j.-re 4 pont

A j. lap mindenképpen a tömb elejére kerül 1 pont

Ha nem volt kézben, akkor a legutolsó lapot teszi le 1 pont

Ha nem volt kézben, akkor minden lap eggyel hátrább kerül 1 pont

Ha kézben volt, akkor az előtte levők mindegyike eggyel hátrább kerül 1 pont

B. A legkevesebbet használt lapját cseréli ki a j.-re 4 pont
Ha nem volt kézben, akkor a legutolsót teszi le 1 pont
Ha nem volt kézben, akkor utolsónak kerül be 1 pont
Ha kézben volt, akkor a használatszám szerinti helyre kerül a tömbben 2 pont
C. A legrégebben bekerült lapját cseréli ki a j.-re 4 pont
Ha nem volt kézben, akkor leteszi a legutolsó lapot 1 pont
Ha nem volt kézben, akkor a többi lapot eggyel hátrább teszi 1 pont
Ha nem volt kézben, akkor az 1. helyre teszi 1 pont
Ha kézben volt, akkor semmit nem tesz 1 pont

3. feladat: Szöveg (10 pont)

A. Elejéről és a végéről felváltva, befelé haladva adja a betűket 8 pont
Részpontszámok adhatók:
Először az első karaktert, 2 pont
aztán az utolsót, 2 pont
ezt követően a másodikat, 1 pont
majd az utolsó előttit, 1 pont
B. AALM 2 pont

4. feladat: Telefonszámok (9 pont)

A. nem nyomnak le PAUSE gombot 1 pont
ekkor az első ciklus végtelen lesz 1 pont
B. nem nyomnak le AUTO gombot 1 pont
ekkor a második ciklus végtelen lesz 1 pont
C. 6-jegyűnél (+ a körzetszám, ha van) rövidebb számot használnak 1 pont
D. 7-jegyűnél (+ a körzetszám, ha van) hosszabb számot használnak 1 pont
E. 06 után hiányzik a körzetszám 1 pont
F. az AUTO gomb után nem számjegyet nyomnak le 1 pont
G. a számjegyek közé más gombokat kevernek 1 pont

5. feladat: Transzformáció (8 pont)

A. FORGAT((1,1),(2,2)) 2 pont
XTÜKÖR((2,1),(3,2)) 2 pont
B. YTÜKÖR((1,1),(3,2)) 2 pont
FORGAT((2,2),(3,3)) 2 pont

6. feladat: Keresés (23 pont)

Ez az algoritmus Knuth-Morris-Pratt algoritmusa, amely B karaktereit visszafelé hasonlítva keresi B-t A-ban, s ha eltérést talál, akkor az eltérés helyétől függő lépésszámmal lép tovább A-ban a következő hasonlításhoz.

A. Ha megtalálta a keresett B szöveget A-ban vagy 3 pont
ha az A i. karakterétől kezdve már nincs B elemszámnyi jel 3 pont
B. Ha B-ben visszafelé haladva megtalálja az első eltérő karaktert vagy 2 pont

ha a hasonlítás során B valamennyi karakterénél azonosságot talált 2 pont

C. C(j)-vel lehet továbblépni A-ban, ha B j. pozícióján volt először eltérés 3 pont

D. A C vektor hátulról monoton növekedő (C(j)≥C(j+1)) 2 pont

C(j)=hossz(B)–j+1, ha B(j) mögött nincs vele azonos betű B-ben C(j+1) hellyel vagy annál többel mögötte 4 pont

C(j)=a C(j+1) hellyel vagy annál többel mögötte található első, B(j)-vel azonos betű távolsága a j. helytől 4 pont

7. feladat: Titkosítás (16 pont)

A. A nagybetűket eltolással titkosítja a Kulcs betűi alapján:

a karakter kódjához hozzáadja a Kulcs következő betűjének kódját, 4 pont

ebből levonja az A-betű kódjának kétszeresét, 2 pont

s végül hozzáad 1-et, 1 pont

az így kapott érték MOD 26 lesz az új betű sorszáma (A=0, B=1, ...) 3 pont

A többi karakterrel nem csinál semmit. 2 pont

B. EZT NEM NEHEZ 4 pont

8. feladat: Csere (10 pont)

A. Ahány eleme van a tömbnek, azaz N-szer 3 pont

B. Az első K vagy N-K elemet cseréli fel az utolsó K vagy N-K elemmel 2 pont

a kettő közül (K,N-K) a kisebbiket 1 pont

C. Az `if K-E>V-K` sorban a feltétel helyesen: `K-E<V-K` 4 pont

1995. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Kocka (16 pont)

A megoldáshoz a kocka minden csúcsához meg kell adni, hogy melyek a szomszédai:

```
szomszéd=({2,4,8},{1,3,7},{2,4,6},{1,3,5},{4,6,8},{3,5,7},
          {2,6,8},{1,5,7})
```

Ezek után a teendő: végig kell menni a beolvasott 8 csúcson (legyenek a `csúcs` tömbben), s mindegyikre megnézni, hogy az előzővel szomszédos-e vagy sem:

```
jó:=igaz
Ciklus i=2-től 8-ig
  Ha csúcs(i)∉szomszéd(csúcs(i-1)))
      akkor jó:=hamis; Ki: csúcs(i-1),csúcs(i)
Ciklus vége
Ha jó akkor Ki: „Bejárható”
```

Értékelési szempontok

Az útmutatóban aláhúzással jelöljük azokat a párokat a felsorolásban, amelyeket nem köt össze él, illetve az utolsó tesztesetben kétszeres aláhúzással szedtük azokat, amelyeket él köt össze.

A. Bejáráskor sorszámnövekedést megenged (1,2,3,4,5,6,7,8) 2 pont

4 után 1, 8 után 5 következhet (3,4,1,2,7,8,5,6) 1 pont

B. Bejáráskor sorszámcsökkenést megenged (8,7,6,5,4,3,2,1) 2 pont

1 után 4, 5 után 8 következhet (6,5,8,7,2,1,4,3) 1 pont

C. Bejáráskor más változást nem enged meg (1,2,4,5,8,7,6,3) 3 pont

D. Függőleges lépéskor 9–X alakú változást megenged (1,8,7,2,3,6,5,4) 3 pont

E. Függőleges lépéskor más változást nem enged meg (1,2,3,4,6,7,8,5) 1 pont

F. Függőleges lépéskor más változást nem enged meg (1,2,3,6,5,8,7,4) 1 pont

G. Minden hibát észrevesz egyetlen felsorolásban (1,3,2,4,8,6,7,5) 2 pont

2. feladat: Metró (32 pont)

Készítsünk egy konstans tömböt a metróállomásokról:

```
Metro=(('Vörösmarty_tér','Deák_tér',itt a többi állomás,
'Mexikói_út','','','','','','','','','','',''),
 ('Déli_pályaudvar',itt a többi állomás,'Örs_vezér_tere','','',
'','','','','','','',''),
('Kőbánya-Kispest',itt a többi állomás,'Újpest-központ'));
```

Másik tömbben tároljuk az átszállóhely sorszámát, valamint az állomások számát:

```
Át=(2,5,11)
Db=(11,11,20)
```

A program a beolvasott két állomásnevet megkeresi a metróállomások tömbjében (melyik vonal hányadik állomása), majd attól függően, hogy azonos vonalon vannak-e, illetve milyen irányban, más-más kiírást alkalmaz:

```
Keresés(Első,i,j); Keresés(Második,k,l)
Ha i=k akkor Kiírás(i,j,k,l)
különben ha Első='Deák_tér' akkor Kiírás(k,Át(k),k,l)
különben ha Második='Deák_tér' akkor Kiírás(i,j,i,Át(i))
különben Kiírás(i,j,i,Át(i)); Ki: 'Átszállás'
         Kiírás(k,Át(k),k,l)
```

A keresés a legelső vonalat keresi meg, ahol a metróállomás megtalálható:

```
Keresés(Mit,i,j)
  i:=1; j:=1; Van:=Metro(1,1)=Mit
  Ciklus amíg nem Van
    j:=j+1; Ha j>db(i) akkor i:=i+1; j:=1
    Van:=Metro(i,j)=Mit
  Ciklus vége
Eljárás vége.
```

```
Kiírás(i,j,k,l)
  Ki: 'Az ',i,'. metróvonalon kell utazni '
  Ha j<l akkor Ki: Metro(i,db(i)),' felé '
      különben Ki: Metro(i,1),' felé '
  Ki: abs(l-j),' állomást.'
Eljárás vége.
```

Értékelési szempontok

A METRO.xxx állományokban valódi budapesti állomások nevét adjuk meg (szóköz helyett aláhúzás jellel). (Az útmutatóban N az N-edik állomást jelenti az állomások felsorolásának sorrendjében, vN pedig ugyanezt, de visszafelé haladva.)

A. Azonos a kezdő- és a végállomás (Népstadion, Népstadion→0) 2 pont

B. A kezdőállomásról előre kell menni (Déli_pályaudvar, Népstadion→8) 6 pont

C. A kezdőállomásról visszafelé kell menni (Árpád_híd, Kálvin_tér→v7) 6 pont

D. Indulás átszállási helyről, a cél a 3. vonalon van (Deák_tér,Lehel_tér→3) 6 pont

E. Átszállásig előre, s utána is előre (Déli_pályaudvar, Hősök_tere→4,7) 3 pont

F. Átszállásig előre, s utána visszafelé (Klinikák, Moszkva_tér→4,v3) 3 pont

G. Átszállásig visszafelé, s utána előre (Népstadion, Hősök_tere→v4,7) 3 pont

H. Átszállásig visszafelé, s utána is visszafelé (Népstadion, Klinikák→v4,v4) 3 pont

3. feladat: Állatok (27 pont)

Az `Eszi` mátrix első oszlopa tartalmazza az evőt, a második oszlopa pedig azt, amit eszik az első oszlopban levő párja. A feladat szerint olyan első oszlopbelieket kell kiírni (azaz állatokat), amelyek esznek állatot (azaz a második oszlopbeli valamelyik párjuk szerepel valahol az első oszlopban is).

A megoldás két lépésben végezhető el. Először meghatározzuk az állatokat (amelyik többször szerepel a táblázatban, azt is csak egyszer):

```
db:=0
Ciklus i=1-től N-ig
  j:=1
  Ciklus amíg j≤db és Evő(j)≠Eszi(i,1)
    j:=j+1
  Ciklus vége
  Ha j>db akkor db:=db+1; Evő(db):=Eszi(i,1)
Ciklus vége
```

Ezután pedig mindegyikről eldöntjük, hogy eszik-e állatot:

```
Ciklus i=1-től db-ig
  j:=1; jó:=hamis
  Ciklus amíg j≤n és nem jó
    Ha Evő(i)=Eszi(j,1) és Állat(Eszi(j,2))
       akkor Ki: Evő(i); jó:=igaz
    különben j:=j+1
  Ciklus vége
Ciklus vége
```

Az `Eszi(j,2)` akkor állat, ha benne van az evők tömbjében:

```
Állat(v):
  k:=1
  Ciklus amíg k≤db és v≠Evő(k)
    k:=k+1
  Ciklus vége
  Állat:=(k≤db)
Függvény vége.
```

Ha már ilyen függvényünk van, akkor felhasználásával az első programrészt is újraírhatjuk:

```
db:=0
Ciklus i=1-től N-ig
  Ha nem Állat(Eszi(i,1)) akkor db:=db+1; Evő(db):=Eszi(i,1)
Ciklus vége
```

Értékelési szempontok

A. Nincs húsevő 3 pont

B. Két evési kapcsolat, az egyik húsevő 3 pont

C. Vannak húsevők, s mindegyik egyszer, legalább egyet megad 3 pont

D. Vannak húsevők, s mindegyik egyszer, mind megadja 6 pont

E. Van többször felsorolt húsevő is 6 pont

F. Van mindenevő (növény- és húsevő) is, ezeket is megadja 6 pont

Kilencedik-tizedik osztályosok

1. feladat: Fenyőfa (20 pont)

A fenyőfa magassága az egymásbaágyazott zárójelek legnagyobb mélysége lesz. A fa tömege a leírásban szereplő F betűk száma, az elágazások száma pedig a nyitózárójelek számának kétszerese:

```
tömeg:=0; nyit:=0; mag:=1; maxmag:=1
Ciklus i=1-től hossz(fa)-ig
  Ha fa(i)∈{'F','f'} akkor tömeg:=tömeg+1
  különben ha fa(i)='(' akkor nyit:=nyit+1; mag:=mag+1
                             ha maxmag<mag akkor maxmag:=mag
  különben mag:=mag-1
Ciklus vége
Ki: maxmag, tömeg, nyit/2
```

Értékelési szempontok

A. Ágnélküli fa [F] helyes magasságú (1) 1 pont

B. Kétágú fa [(F)F(F)] helyes magasságú (2) 2 pont

C. Balra eggyel hosszabb fa [(((F)F(F))F(F))F((F)F(F))] jó magasságú (4) 3 pont

D. Jobbra eggyel hosszabb fa [((F)F(F))F((F)F((F)F(F)))] jó magasságú (4) 3 pont

E. Ágnélküli fa [F] helyes tömegű (1) 1 pont

F. Kétágú fa [(F)F(F)] helyes tömegű (3) 1 pont

G. Ágas fa [(((F)F(F))F(F))F((F)F(F))] helyes tömegű (9) 3 pont

H. Ágnélküli fa [F] helyes elágazásszámú (0) 1 pont

I. Egy elágazásos fa [(F)F(F)] helyes elágazásszámú (1) 2 pont

J. Több elágazásos fa [(((F)F(F))F(F))F((F)F(F))] helyes elágazásszámú (4) 3 pont

2. feladat: Televízió (20 pont)

Első lépésként a beolvasott adatokat alakítsuk át: az `adó(i,j)` értéke legyen 1, ha az i-edik adón a j-edik órában van adás, s közben számoljuk az adók számát is:

```
n:=0
Ciklus amíg nem vége(f)
  Olvas(f,asz,ak,av); Ha asz>n akkor n:=asz
  Ciklus i=ak-tól av-ig
    adó(asz,i):=1
  Ciklus vége
Ciklus vége
```

Ezzel kaptunk N darab intervallumsorozatot. Először számoljuk meg, hogy melyik órában hány adás van:

```
Ciklus i=0-tól 23-ig
  st(i):=0
Ciklus vége
Ciklus i=0-tól 23-ig
  Ciklus j=1-től n-ig
    st(i):=st(i)+adó(j,i)
  Ciklus vége
Ciklus vége
```

Az A részfeladat megoldásához vegyük ebből a leghosszabb 0-sorozatot:

```
i:=1; maxnullk:=0; maxnullv:=0
Ciklus amíg i<24
  Ha st(i)=0 akkor nullk:=i
                   Ciklus amíg i<24 és st(i)=0
                     i:=i+1
                   Ciklus vége
                   Ha i<24 akkor nullv:=i különben nullv:=23
                   Ha nullv-nullk>maxnullv-maxnullk
                      akkor maxnullk:=nullk; maxnullv:=nullv
  Elágazás vége
Ciklus vége
Ha maxnullk>0 akkor Ki: maxnullk,maxnullv
```

A B részfeladat megoldása sokkal egyszerűbb, az `st` tömb maximális értékű elemei indexét kell kiírni:

```
maxad:=st(0)
Ciklus i=1-től 23-ig
  Ha st(i)>maxad akkor maxad:=st(i)
Ciklus vége
Ki: maxad
Ciklus i=0-tól 23-ig
  Ha st(i)=maxad akkor Ki: i,i+1
Ciklus vége
```

Értékelési szempontok

A. Üres állományra jó	1 pont
B. Egy adás esetén jó a leghosszabb adásszünet kiválasztása	2 pont
C. Egy adó, több adás esetén jó a leghosszabb adásszünet kiválasztása	2 pont
D. Több adó esetén jó a leghosszabb adásszünet kiválasztása	4 pont
E. Nap elején jó a leghosszabb adásszünet kiválasztása	1 pont
F. Nap végén jó a leghosszabb adásszünet kiválasztása	1 pont
G. Ha nincs adásszünet, akkor ezt is megadja	2 pont
H. Egy adó esetén jó a legtöbb adást tartalmazó időszak meghatározása (időszak, darabszám)	1-1 pont
I. Több adó esetén jó a legtöbb adást tartalmazó időszak meghatározása (időszak, darabszám)	3-2 pont

3. feladat: Metró (20 pont)

A feladat abban különbözik az 5-8. osztályosok feladatától, hogy az átszállóhely nem rögzített, illetve nem garantált, hogy az átszállóhely a 3 metróvonal közös állomása. Készítsünk itt is egy konstans tömböt a metróállomásokról:

```
Metro= (('Vörösmarty_tér','Deák_tér',itt a többi állomás, 'Mexikói_út',
'','','','','','','','','',''),
('Déli_pályaudvar',itt a többi állomás,'Örs_vezér_tere','','',
'','','','','','','',''),
('Kőbánya-Kispest',itt a többi állomás,'Újpest-központ'));
```

Másik tömbben tároljuk az állomások számát:

```
Db=(11,11,20)
```

A program a beolvasott két állomásnevet megkeresi a metróállomások tömbjében (melyik vonal hányadik állomása), majd attól függően, hogy azonos vonalon vannak-e, illetve milyen irányban, más-más kiírást alkalmaz:

```
Átszállóhelykeresés
Keresés(Első,i,j); Keresés(Második,k,l)
Ha i=k és j=l akkor Ki: 'Nem kell utazni'
különben ha i=k akkor Kiírás(i,j,k,l)
különben ha Át(i,k)≠0 akkor Átszállás(i,j,k,l)
különben m:=1; ha i=m vagy k=m akkor m:=m+1
         ha i=m vagy k=m akkor m:=m+1
         Átszállás(i,j,m,Át(i,m)); Átszállás(m,Át(m,j),k,l)
```

Ha különböző valon vannak, akkor átszállásra van szükség. A fenti algoritmusban már eldőlt, hogy az átszállást egyszer vagy kétszer kelle-e meghívni.

```
Átszállás(i,j,k,l):
  Ha Első=Metro(i,Át(i,k)) akkor Kiírás(k,Át(k,i),k,l)
  különben ha Második=Metro(k,Át(i,k))
              akkor Kiírás(i,j,i,Át(i,k))
           különben Kiírás(i,j,i,Át(i,k))
                    Kiírás(k,Át(k,i),k,l)
Eljárás vége.
```

A keresés a legelső vonalat keresi meg, ahol a metróállomás megtalálható:

```
Keresés(Mit,i,j)
  i:=1; j:=1; Van:=Metro(1,1)=Mit
  Ciklus amíg nem Van
    j:=j+1; Ha j>db(i) akkor i:=i+1; j:=1
    Van:=Metro(i,j)=Mit
  Ciklus vége
Eljárás vége.
```

Meg kell keresni az egyes vonalak közötti átszállóhelyeket:

```
Átszállóhelykeresés:
  Ciklus ki=1-től Metrószám-1-ig
    Ciklus kj=ki+1-től Metrószám-ig
      Át(ki,kj):=0; Át(kj,ki):=0; Van:=hamis; i:=1
      Ciklus amíg i≤db(ki) és nem Van
        j:=1
        Ciklus amíg j≤db(kj) és Metro(ki,i)≠Metro(kj,j)
          j:=j+1
        Ciklus vége
        Ha j≤db(kj) akkor Át(ki,kj):=i; Át(kj,ki):=j
                          Van:=igaz
                   különben i:=i+1
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.

Kiírás(i,j,k,l)
  Ki: 'Az ',i,'. metróvonalon kell utazni '
  Ha j<l akkor Ki: Metro(i,db(i)),' felé '
      különben Ki: Metro(i,1),' felé '
  Ki: abs(l-j),' állomást.'
Eljárás vége.
```

Értékelési szempontok

A METRO1.DAT-ban valódi budapesti állomások nevét adjuk meg (szóköz helyett aláhúzás jellel), kivéve az 1. metróvonal egyik állomását (ez most a Deák_tér helyett az Astoria). (Az útmutatóban N az N-edik állomást jelenti az állomások felsorolásának sorrendjében, vN pedig ugyanezt, de visszafelé haladva.)

A. Átszállás nélkül:

Azonos kezdő és végállomás (Népstadion, Népstadion→0)	1 pont
A kezdőállomásról előre kell menni (Déli_pályaudvar, Népstadion→8)	2 pont
A kezdőállomásról visszafelé kell menni (Árpád_híd, Kálvin_tér→v7)	2 pont
A kezdőállomás átszállási hely, a cél innen elérhető (Astoria, Hősök_tere→7)	2 pont

B. 1 átszállással:

Átszállásig előre, utána is előre (Déli_pályaudvar, Hősök_tere→5,7)	1 pont
Átszállásig előre, s utána visszafelé (Klinikák, Moszkva_tér→4,v3)	1 pont
Átszállásig visszafelé, s utána előre (Népstadion, Hősök_tere→v3,7)	1 pont
Átszállásig visszafelé, s utána is visszafelé (Népstadion, Klinikák→v4,v4)	1 pont
Átszállási helyről indulva (Deák_tér, Opera→1,2)	2 pont

C. 2 átszállással:

Nem átszállási helyről indulva (Opera, Nyugati_pályaudvar→v2,1,2)	3 pont

A METRO2.DAT-ban valódi budapesti állomások nevét adjuk meg (szóköz helyett aláhúzás jellel), azaz egyetlen, közös átszállási hely van, a Deák_tér.

D. Az átszállási helyen a 3. vonalra is át lehet szállni (Oktogon, Kálvin_tér→v3,v2) 2 pont

E. Az átszállási helyről indulva a 3. vonalra is mehetünk (Deák_tér, Ecseri_út→v7) 2 pont

4. feladat: Állatok (15 pont)

Az `Eszi` mátrix első oszlopa tartalmazza az evőt, a második oszlopa pedig azt, amit eszik az első oszlopban levő párja. A feladat szerint olyan első oszlopbelieket kell kiírni (azaz állatokat), amelyek esznek állatot (azaz a második oszlopbeli valamelyik párjuk szerepel valahol az első oszlopban is).

A megoldás két lépésben végezhető el. Először meghatározzuk az állatokat (amelyik többször szerepel a táblázatban, azt is csak egyszer):

```
db:=0
Ciklus i=1-től N-ig
  j:=1
  Ciklus amíg j≤db és Evő(j)≠Eszi(i,1)
    j:=j+1
  Ciklus vége
  Ha j>db akkor db:=db+1; Evő(db):=Eszi(i,1)
Ciklus vége
```

Ezután pedig mindegyikről eldöntjük, hogy eszik-e állatot (azt tudjuk, hogy valamit biztosan eszik):

```
Ciklus i=1-től db-ig
  j:=1
  Ciklus amíg j≤N és (Evő(i)≠Eszi(j,1) vagy
                      nem Állat(Eszi(j,2))
    j:=j+1
  Ciklus vége
  Ha j>N akkor Ki: Evő(i)
Ciklus vége
```

Értékelési szempontok

A. Üres állomány	1 pont
B. Egyetlen táplálkozási pár	1 pont
C. Csak növényevők vannak, s mindegyik egyszer	2 pont
D. Van többször felsorolt növényevő is, de csak egyszer írja ki	3 pont
E. Van húsevő is	3 pont
F. Van mindenevő (növény- és húsevő) is	5 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Fenyőfa (15 pont)

A fa magassága = a legmélyebb zárójelezés mélysége + 1.

A fa tömege = a leírásában szereplő F betűk száma $1/2^K$-nal súlyozva, ahol K az adott F betű zárójelezésének mélysége.

A közös elágazásból induló ágak számának maximuma = a szomszédos, azonos zárójelmélységű F-ek száma.

```
tömeg:=0; akttöm:=1; mag:=1; maxmag:=1; elmag:=0; maxág:=0
ágszám:=(0,…,0)
Ciklus i=1-től hossz(fa)-ig
  Ha fa(i)∈{'F','f'} akkor tömeg:=tömeg+akttöm
  különben ha fa(i)='(' akkor mag:=mag+1; akttöm:=akttöm/2
                             ha maxmag<mag akkor maxmag:=mag
                             ágszám(mag):=ágszám(mag)+1
                             Ha ágszám(mag)>maxág
                                akkor maxág:=ágszám(max)
  különben mag:=mag-1; akttöm:=akttöm*2
           ágszám(mag+2):=0
Ciklus vége
Ki: maxmag, tömeg, maxág
```

Értékelési szempontok

A. Ágnélküli fa [F] helyes magasságú (1)	1 pont
B. Kétágú fa [F(F)(F)] helyes magasságú (2)	1 pont
C. Sokágú fa [F(F)(F)(F)(F)] helyes magasságú (2)	1 pont
D. Bal szélén hosszabb fa [F(F(F(F)(F))(F))(F)(F)(F)] jó magasságú (4)	1 pont
E. Középen hosszabb fa [F(F)(F(F)(F(F)(F))(F))(F)(F)] jó magasságú (4)	1 pont
F. Jobb szélén hosszabb fa [F(F)(F)(F(F)(F))] jó magasságú (3)	1 pont
G. Ágnélküli fa [F] helyes tömegű (1)	1 pont
H. Kétágú fa [F(F)(F)] helyes tömegű (2)	1 pont
I. Sokágú fa [F(F)(F)(F)(F)] helyes tömegű (3)	1 pont
J. Magasabb fa [F(F)(F)(F(F)(F))] helyes tömegű (3)	2 pont
K. Ágnélküli fa [F] maximális ágszáma helyes (0)	1 pont
L. Egyelágazásos fa [F(F)(F)(F)(F)] maximális ágszáma helyes (4)	1 pont
M. Többelágazásos fa [F(F)(F(F)(F(F)(F)(F))(F))] maximális ágszáma helyes (3)	2 pont

2. feladat: Televízió (16 pont)

A feladat nagyon hasonlít a 9-10. osztályosok feladatára, azzal a különbséggel, hogy több napot is kell vizsgálni, valamint a perceket is figyelembe kell venni. Ezek miatt a megoldásban használt adatszerkezet alapvetően eltér az ott leírt változattól, itt az intervallumokat kezdő- és végpontjukkal kell tárolni (nap, óra perc). Az `Adás 2*M` elemű tömbben legyenek az egyes műsorok kezdő- és végidőpontjai! Rendezzük sorba, jelölve a kezdő- és a végidőpontokat is! Szúrjuk be közéjük az N nap végidőpontjait is (i,24,0)!

Az A feladat megoldása: Vegyük az egyes időpontokat, a kezdet esetén egy számlálót növeljünk eggyel, vég esetén pedig csökkentsünk! A feladat megoldás a leghosszabb szakasz, amikor a számláló 0 értékű (figyelve arra, hogy egy napon belüli legyen)!

```
Max:=0; Maxk:=(0,0,0); Maxv:=(0,0,0); k:=(1,0,0); db:=0
Ciklus i=1-től 2*M+N-ig
  Ha Adás(i).típus=kezd
     akkor Ha db=0 és Adás(i).idő-k>Max
              akkor Max:=Adás(i).idő-k[1]; Maxk:=k
                    Maxv:=Adás(i).kezd
           Elágazás vége
           db:=db+1
  különben Ha Adás(i).típus=vég
              akkor db:=db-1; Ha db=0 akkor k:=Adás(i).idő+1[2]
           Elágazás vége
  különben {Nap vége}
           Ha db=0 és Adás(i).idő-k>Max
              akkor Max:=Adás(i).idő-k; Maxk:=k
                    Maxv:=Adás(i).kezd
           Elágazás vége
           Ha db=0 akkor k:=Adás(i).idő+1
  Elágazás vége
Ciklus vége
```

A B részfeladat hasonló ötletre épül, de ezelőtt az időpontokat „kerekíteni” kell, azaz el kell hagyni a percet, a végidőpontoknál azonban az órát meg kell növelni eggyel. Ezután ugyanúgy kell számolni az éppen futó adások számát, mint az A részfeladatban, majd a maximális darabszámú adásórákat kell megadni eredményül (itt a nap végének nincs szerepe):

```
maxdb:=0; db:=0; adb:=0
Ciklus i=1-től 2*M-ig
  Ha Adás(i).típus=kezd
     akkor db:=db+1
           Ha db>maxdb akkor maxdb:=db; adb:=1; a(edb):=i
           különben ha db=maxdb akkor adb:=adb+1; a(edb):=i
  különben db:=db-1
Ciklus vége
```

Értékelési szempontok

A. Üres állományra jó	1 pont
B. Egy adás esetén jó a leghosszabb adásszünet kiválasztása	1 pont
C. Egy adó, több adás, egy nap esetén jó a leghosszabb adásszünet kiválasztása	1 pont
D. Egy adó, több adás, több nap esetén jó a leghosszabb adásszünet kiválasztása	1 pont

[1] Két idő típusú adat (nap,óra,perc) különbsége.
[2] Az időt megnöveli a következő napra, azaz (i,24,0)-ból (i+1,0,0)-t csinál.

E. Több adó esetén jó a leghosszabb adásszünet kiválasztása 3 pont

F. A perceket is jól számolja 2 pont

G. Ha nincs adásszünet, akkor is jó eredmény ad 2 pont

H. Egy adó esetén jó a legtöbb adást tartalmazó perc megadása (időpont, darabszám) 1 pont

I. Több adó, egy nap esetén jó a legtöbb adást tartalmazó perc megadása (időpont, darabszám) 1-1 pont

J. Több adó, több nap esetén jó a legtöbb adást tartalmazó perc megadása (időpont, darabszám) 1-1 pont

3. feladat: Sokszög (12 pont)

Első lépésként sorba kell rendezni a pontokat x-, azon belül pedig y-koordináta szerint:

```
Rendezés(P,N):
  Ciklus i=1-től N-1-ig
    min:=i
    Ciklus j=i+1-től N-ig
      Ha P(i).x<P(min).x vagy
         P(i).x=P(min).x és P(i).y=P(min).y akkor min:=i
    Ciklus vége
    Csere(P(i),P(min))
  Ciklus vége
Eljárás vége.
```

Ezután a 2..N pontokat sorba kell rendezni az első ponttal összekötő egyenesük iránytangense szerint növekvő sorrendben. Tudjuk, hogy $szög(P_1,P_i) < szög(P_1,P_j)$ akkor és csak akkor, ha

$$\tan(szög(P_1,P_i)) = \frac{P_i.y - P_1.y}{P_i.x - P_1.x} < \frac{P_j.y - P_1.y}{P_j.x - P_1.x} = \tan(szög(P_1,P_j)).$$

Mivel az első pont a legkisebb, ezért a képletben szereplő nevezők egyike sem lehet negatív, így a reláció mindkét oldala szorozható velük, azaz $szög(P_1,P_i) < szög(P_1,P_j)$ akkor és csak akkor, ha $(P_i.y - P_1.y) * (P_j.x - P_1.x) < (P_j.y - P_1.y) * (P_i.x - P_1.x)$

```
Rendezés2(P,N):
  Ciklus i=2-től N-1-ig
    min:=i
    Ciklus j=i+1-től N-ig
      Ha (P(j).y-P(1).y)*(P(min).x-P(1).x)<
         (P(j).x-P(1).x)*(P(min).y-P(1).y) akkor min:=i
    Ciklus vége
    Csere(P(i),P(min))
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A. Minden pont a kiindulópont felett található (jó a kezdőpont, a sorrend, az irány) 1-1-1 pont

B. Minden pont a kiindulópont alatt található (jó a sorrend, az irány) 2-1 pont

C. A kiindulópont felett és alatt is vannak pontok (jó a sorrend, az irány) 2-1 pont

D. Több (2) legkisebb x-koordinátájú pont van (jó a kezdőpont, a sorrend, az irány) 1-1-1 pont

4. feladat: Kutya (12 pont)

Legyen a kutya a (0,0) koordinátájú pontban! Ekkor a gazda a feladat szövege szerint az (a,b) koordinátájú pontban van. A feladat megoldása arra az esetre, ha a kutya sebessége (C) nem nulla):

```
K:=(0,0); G:=(a,b); idő:=0
Ciklus amíg b-K.y≥1
  Ha G.x=K.x akkor Irány:=0
          különben Irány:=arctan((G.y-K.y)/(G.x-K.x))
  dx:=(C*cos(Irány)+D)*E; dy:=C*sin(Irány)*E
  K.x:=K.x+dx; K.y:=K.y+dy; idő:=idő+1
Ciklus vége
táv:=a-K.x
```

Értékelési szempontok

A. A fenti példát jól oldja meg	1 pont
B. Szélesebb folyóban lassabban ér célba	1 pont
C. Messzebb levő gazda esetén hamarabb odaér (nem kell visszafelé úsznia)	1 pont
D. Lassúbb kutya lassabban ér oda	1 pont
E. Gyorsabb folyó esetén a kutya lassabban ér oda	1 pont
F. Nagyobb időintervallum, pontatlanabb közelítés	1 pont
G. Ha eleve felfelé kell úsznia, sokkal lassabban ér oda	1 pont
H. Feleakkora folyósebesség mellett sokkal gyorsabban ér célba	1 pont
I. 0 szélességű folyó	1 pont
J. A gazda pontosan szemben van	1 pont
K. 0 sebességű kutya sohasem ér célba	1 pont
L. Álló vízben nagyon gyorsan célba ér	1 pont

5. feladat: Automata (20 pont)

A szalag jelei a `JEL(40)` tömbben legyenek, a szabályok pedig a `SZABALY(N)` vektorban! A feladat megoldása:

```
Hely:=20; Allapot:=1; idő:=0; vége:=hamis
Ciklus amíg idő≤1000 és nem vége
  Keresés(Allapot,JEL(Hely),i,van)
  Ha van akkor Allapot:=SZABALY(i).a2
               Jel(Hely):=SZABALY(i).j2
               Ha SZABALY(i).betű='B' akkor Balralép
               Ha SZABALY(i).betű='J' akkor Jobbralép
       különben Ki: 'NEMDEFINIÁLT'; Ki: Allapot,JEL(Hely)
               vége:=igaz
  idő:=idő+1
Ciklus vége
Ha nem vége akkor Ha idő>1000 akkor Ki: 'VÉGTELEN'
                              különben Ki: 'VÉGES'; Ki: JEL
```

A balra, illetve a jobbra lépésnél nézni kell, hogy a fej elhagyja-e a szalagot:

```
Balralép:
  Ha Hely=1 akkor Ki: 'HIBA, BALRA KILÉPETT'
          különben Hely:=Hely-1
Eljárás vége.
```

```
Jobbralép:
  Ha Hely=40 akkor Ki: 'HIBA, JOBBRA KILÉPETT'
         különben Hely:=Hely+1
Eljárás vége.
```

```
Keresés(a,j,i,van):
  i:=1
  Ciklus amíg i≤N és nem(SZABALY(i).a1=a és SZABALY(i).j1=j)
    i:=i+1
  Ciklus vége
  van:=i≤N
Eljárás vége.
```

Értékelési szempontok

A. 1 állapot, VEGES
####################00000################ 2 pont

B. 1 állapot, HIBA, BALRA KILEPETT 1 pont

C. 1 állapot, VEGTELEN 3 pont

D. 1 állapot, HIBA, JOBBRA KILEPETT 1 pont

E. 2 állapot, NEMDEFINIALT, 2 # 3 pont

F. 1 állapot, VEGES
####################1010################ 3 pont

G. 2 állapot, VEGES
####################10101################ 3 pont

H. Sok állapot, VEGES
####################0000000############ 4 pont

1995. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mondd ki a számokat (23 pont)

A megoldást könnyű elkészíteni, ha tároljuk egy-egy konstans tömbben az egyesek és a tízesek nevét:

```
tizesek=('tíz','húsz','harminc','negyven','ötven','hat-
van','hetven','nyolcvan','kilencven','száz')
egyesek=('egy','kettő','három','négy','öt','hat','hét',
'nyolc','kilenc')
```

Ekkor egy X szám kiírása a következő lehet:

```
j:=X div 10; k:=X mod 10
Ha j>0 akkor Ha j=1 és k>0 akkor Ki: 'tizen'
             különben ha j=2 és k>0 akkor Ki: 'huszon'
             különben Ki: tizesek(j)
Ha k>0 akkor Ki: egyesek(k)
```

Értékelési szempontok

A. Van beolvasás 1 pont

B. Van valamilyen kiírás 1 pont

C. Egyjegyű számokra helyes (jó számonként 0.5 pont felfelé kerekítve) 5 pont

D. Nullára végződőekre helyes (jó számonként 0.5 pont felfelé kerekítve) 5 pont

E. Tizen..-, huszon..-ra helyes (jól kezdődik 1-1 pont, jól folytatódik 0.5-0.5 pont) 3 pont

F. Harminc..-, ..., kilencven..-re helyes 6 pont

(maximum 3 pont, ha csak a 10-es helyiérték jó,

maximum 3 pont, ha csak az 1-es helyiérték jó)

G. Százra helyes 2 pont

2. feladat: Autóbusz-menetrend (25 pont)

Tartalmazza a `Várakozás(N)` tömb az egyes állomásokon várakozási időket, a `Menetidő(N)` tömb pedig a menetidőket! Az első állomáson nincs várakozási idő! Az `óra` és a `perc` változók az indulási időt tartalmazzák.

```
Ciklus i=1-től n-1-ig
  Ha i>1 akkor perc:=perc+várakozás(i)
  perc:=perc+menetidő(i)
  Ha perc>59 akkor óra:=óra+1; perc:=perc-60
  Ki: i,óra,perc
Ciklus vége
```

Értékelési szempontok

A. N beolvasása 1 pont

B. N–2 várakozási idő beolvasása 1 pont

C. N–1 menetidő beolvasása 1 pont

D. jól kiírja az indulási időt 1 pont

E. valamilyen értékű N–1 érkezési idő kiírása 1 pont

F. az 1. állomásra érkezés ideje = indulási idő + menetidő 5 pont

G. a többi állomásra érkezés ideje = előző érkezési idő + várakozási idő + menetidő 10 pont

(–5 pont, ha nem adja hozzá a várakozási időt)

H. Észreveszi az átvitelt percről órára 2 pont

I. Az órát ilyenkor növeli 1-gyel 1 pont

J. A percet ilyenkor csökkenti 60-nal 2 pont

3. feladat: Lóverseny (15 pont)

A megoldásban az `erős(N)` tömb tartalmazza az egyes lovak erősségét.

```
Ciklus i=1-től N-ig
  Ha erős(i)>90 akkor Ki: i,45
  különben ha erős(i)≤50 akkor Ki: i,0
  különben Ki: i,((erős(i)-46) div 5)*5
Ciklus vége
```

Értékelési szempontok

A. Adatok beolvasása (N, erősségek) 1+1 pont

B. Minden lóra valamilyen indulási hely kiírása 1 pont

C. 50-nél gyengébbek a rajtvonalról indulnak 3 pont

(–1 pont, ha az 50-es erősségű nem onnan indul)

D. 90-nél erősebbek a rajtvonaltól 45 méterre indulnak 3 pont

(–1 pont, ha a 90-es erősségű is onnan indul)

E. 51 és 90 közöttiekre jól számol távolságot 6 pont

(–2 pont, ha 1 erősséggel elcsúszik a távolsághozzárendelés)

4. feladat: Furcsa egyszerűsítés (12 pont)

Az összes lehetséges A,B,C,D,E számötöst meg kell vizsgálni:

```
Ciklus a=1-től 9-ig
  Ciklus b=0-tól 9-ig
    Ha b≠a akkor
      Ciklus c=1-től 9-ig
        Ciklus d=0-tól 9-ig
          Ha d≠a és d≠b akkor
            Ciklus e=0-tól 9-ig
              Ha e≠a és e≠b és e≠d akkor
                Ha (100*a+10*b+c)*(10*d+e)=
                                  (10*a+b)*(100*c+10*d+e)
                   Ki: 100*a+10*b+c,100*c+10*d+e
            Ciklus vége
        Ciklus vége
      Ciklus vége
  Ciklus vége
Ciklus vége
```

Értékelési szempontok

A. Mindkét jó számpárt megadja (217, 775; 249,996)		12 pont
Ha csak az egyiket adja meg	6 pont	
Ha A=0-t is előállít	–2 pont	
Ha egy rossz megoldást ad (A≠0)	–1 pont	
Ha több rossz megoldást ad (A≠0)	–2 pont	
A, B, D, E nem mind különbözőek (egyenlőségenként –1 pont)	–6 pont	

Kilencedik-tizedik osztályosok

1. feladat: MIDI (25 pont)

Mivel a bemenő file-ban több MIDI program is lehet, ezért a beolvasás közben kell a feldolgozást elvégezni. Az alábbi algoritmusrészlet onnan indul, hogy beolvastuk az `idő`, `beki` és `magasság` tömbökbe a következő MIDI programot. Legyen a `hang` tömb i-edik eleme igaz, ha az adott időpontban az i-edik hang be van kapcsolva, a `kikap` tömb i-edik eleme pedig az utolsó kikapcsolás ideje!

```
i:=1
Ciklus amíg i≤db
  Ha beki(i)='ON' és hang(magasság(i)) akkor aeset(i)
  különben ha beki(i)='ON' és kikap(magasság(i))=idő(i)
              akkor beset(i)
  különben ha beki(i)='OFF'
              akkor kikap(magasság(i)):=idő(i)
                     hang(magasság(i)):=hamis
  különben ha beki(i)='ON' akkor hang(magasság(i)):=igaz
  Elágazás vége
  i:=i+1
Ciklus vége
```

A c) feltétel szerint, ha valamelyik hang a program végén még szól, akkor azt ki kell kapcsolni (azaz a program végére kell írni további sorokat):

```
utolsó:=idő(db)
Ciklus i=1-től 127-ig
  Ha hang(i) akkor db:=db+1; idő(db):=utolsó+1
                  beki(db):='OFF'; magasság(db):=i
Ciklus vége
```

Ha még szóló hangot akarunk újra megszólaltatni, akkor előtte ki kell kapcsolni, az utána levő felesleges kikapcsolást pedig el kell tüntetni:

```
aeset(i):
  j:=i+1
  Ciklus amíg nem(beki(j)='OFF' és magasság(j)=magasság(i))
    j:=j+1
  Ciklus vége
  Ciklus amíg idő(j-1)>idő(i)-1
    idő(j):=idő(j-1); beki(j):=beki(j-1)
    magasság(j):=magasság(j-1); j:=j-1
  Ciklus vége
  idő(j):=idő(i+1)-1; beki(j):='OFF'
  magasság(j):=magasság(i+1); hang(magasság(i+1)):=igaz
  i:=i+1
Eljárás vége.
```

Az ugyanazon időponthoz tartozó ON és OFF parancsok közül az OFF parancsot 1 időegységgel hamarabb kell kiadni:

```
beset(i):
  j:=i-1
  Ciklus amíg nem(beki[j]='OFF' és magasság(j)=magasság(i))
    j:=j-1
  Ciklus vége
  Ciklus amíg idő(j-1)=idő(i)
    idő(j):=idő(j-1); beki(j):=beki(j-1)
    magasság(j):=magasság(j-1); j:=j-1
  Ciklus vége
  idő(j):=idő(i)-1; beki(j):='OFF'
  magasság(j):=magasság(i); hang(magasság(i)):=igaz
Eljárás vége.
```

Értékelési szempontok

A. Átfedő hang esetén az első OFF-ot a második ON elé teszi	4 pont
1 időegységgel korábbra állítva	2 pont
a régi helyéről megszünteti	2 pont
akkor is jó, ha volt közöttük valami más is	2 pont
B. Ugyanazon időpontú ON és OFF sorrendjét felcseréli, ha kell	2 pont
az OFF-ot 1 időegységgel előbbre teszi	2 pont
akkor is cserél, ha volt közöttük más	2 pont
akkor is előbbre viszi, ha volt közöttük más	2 pont
C. A futás végén a megmaradt hangokat kikapcsolja	2 pont
D. Minden más parancsot meghagy	3 pont
E. Felismeri a program végét	1 pont

F. Felismeri az állomány végét 1 pont

2. feladat: Kirándulók (50 pont)

A `szoba` tömb tartalmazza az egyes szobák befogadóképességét, a `lakószám` tömbbe az egyes szobákba beosztott emberek száma kerül, a `szobája` tömb i-edik eleme pedig azt adja meg, hogy az i-edik turista melyik szobába kerül.

Az első esetben az érkezőket a legelső szobába kell tenni, ahol még van hely:

```
j:=1
Ciklus i=1-től turistaszám-ig
  Ha lakószám(j)=szoba(j) akkor j:=j+1
  lakószám(j):=lakószám(j)+1; szobája(i):=j
Ciklus vége
```

A második esetben először a lányokat osztjuk be, majd új szobát nyitunk, ha az első fiúval kell foglalkoznunk:

```
j:=1
Ciklus i=1-től turistaszám-ig
  Ha lakószám(j)=szoba(j) vagy i=lányszám+1 akkor j:=j+1
  lakószám(j):=lakószám(j)+1; szobája(i):=j
Ciklus vége
```

A harmadik esetben az igények miatt visszalépéses keresést kell alkalmaznunk (sajnos mindenféle stratégiáról be lehet látni, hogy nem eredményez minden esetben megoldást):

```
i:=1
Ciklus amíg i≥1 és i≤lányszám
  Jóesetkeresés(i,j,van)
  Ha van akkor lakószám(j):=lakószám(j)+1; szobája(i):=j
               i:=i+1
       különben lakószám(szobája(i)):=lakószám(szobája(i))-1
               szobája(i):=0; i:=i-1
Ciklus vége
nincsmegoldás:=i<1
```

Az i-edik turistát a j-edik szobába tehetjük, ha befér oda és az igényelt párja nem másik szobában van:

```
Jóesetkeresés(i,j,van):
  j:=szobája(i)+1
  Ciklus amíg j≤szobaszám és
             (lakószám(j)=szoba(j) vagy Rossz(i,j))
    j:=j+1
  Ciklus vége
  van:=j≤szobaszám
Eljárás vége.
```

Két ember egy szobába tehető, ha azonos neműek, illetve különböző szobába tehető, ha nem kérték párnak egymást:

```
Rossz(i,j):
  k:=1
  Ciklus amíg k<i és (j=szobája(k) és (i≤lányszám)=
              (k≤lányszám) vagy j≠szobája(k) és Külön(i,k))
    k:=k+1
  Ciklus vége
  Rossz:=k<i
Függvény vége.
```

Az i-edik és a k-adik külön lehet, ha nem kérték párnak egymást:

```
Külön(i,k):
  l:=1
  Ciklus amíg l≤párszám és nem(ker(l,1)=i és ker(l,2)=k
                          vagy ker(l,2)=i és ker(l,1)=k)
    l:=l+1
  Ciklus vége
  Külön:=l>párszám
Függvény vége.
```

A negyedik részfeladatban azokra is figyelni kell, akik a szobájukba a kérteken kívül mást nem engednének be. Tehát két ember egy szobába tehető, ha együtt akarnak lenni, vagy ha azonos neműek, és nem tiltják ki egymást a szobából.

```
Rossz(i,j):
  k:=1
  Ciklus amíg k<i és (j=szobája(k) és (Akar(i,k) vagy
         nem Tilt(i,k) és (i≤lányszám)=(k≤lányszám)) vagy
         j≠szobája(k) és Külön(i,k))
    k:=k+1
  Ciklus vége
  Rossz:=k<i
Függvény vége.
```

Az i-edik és a k-adik együtt akar lenni, ha ugyanabban a csoportban van:

```
Akar(i,k):
  j:=1
  Ciklus amíg j≤csoportszám és i∉Csoport(j) és k∉Csoport(j)
    j:=j+1
  Ciklus vége
  Akar:=j≤csoportszám és i∈Csoport(j) és k∈Csoport(j)
Függvény vége
```

A k-adik kitiltja az i-ediket, ha ő egy csoportban van és az i-edik nincs abban a csoportban:

```
Tilt(i,k):
  j:=1
  Ciklus amíg j≤csoportszám és k∉Csoport(j)
    j:=j+1
  Ciklus vége
  Tilt:=j≤csoportszám és i∉Csoport(j) és k∈Csoport(j)
Függvény vége
```

Értékelési szempontok

A. Menü		2 pont
B. Eredmény helyes formátumú		3 pont
szobaszám szerint rendezett	1 pont	
szobák lakószámát kiírja	1 pont	
fiúkat és lányokat megkülönbözteti	1 pont	
C. 1. részfeladat: folytonosan feltölti a szobákat		3 pont
optimálisan helyezi el a turistákat		3 pont
észreveszi, ha nincs megoldás		2 pont
D. 2. részfeladat: a fiúkat és a lányokat különválogatja		4 pont
optimálisan helyezi el a turistákat		3 pont

mindenkit elhelyez	1 pont
észreveszi, ha nincs megoldás	2 pont
E. 3. részfeladat: figyeli az egy helyre teendőket	6 pont
optimálisan helyezi el a turistákat	3 pont
mindenkit elhelyez	1 pont
megtartja a 2. részfeladat feltételét	2 pont
észreveszi, ha az egy szobába teendők miatt nincs megoldás	2 pont
F. 4. részfeladat: nem tesz az egy szobába kerülőkön kívül mást a szobájukba	6 pont
mindenkit elhelyez	1 pont
megtartja a 2. részfeladat feltételét (ahol kell)	2 pont
megtartja a 3. részfeladat feltételét	2 pont
észreveszi, ha az egy szobába teendők miatt nincs megoldás	2 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: MIDI (25 pont)

A feladat megoldása lényegében azonos a 9-10. osztályosok feladata megoldásával. Két további teendőnk van. Ha úgy kapcsolnánk ki hangot, hogy nem is szól, akkor a kikapcsolást ki lehet hagyni; ha pedig ugyanabban az időben kétszer kapcsolnánk be, az egyik akkor is lehagyható (dőlten szedtük az újdonságot a fő részben):

```
i:=1
Ciklus amíg i≤db
  Ha beki(i)='ON' és hang(magasság(i))
     akkor aeset(i)
           ha bekap(magasság(i))=idő(i)
              akkor töröl(i); bekap(magasság(i)):=idő(i)
  különben ha beki(i)='ON' és kikap(magasság(i))=idő(i)
            akkor beset(i)
                 ha bekap(magasság(i))=idő(i)
                   akkor töröl(i); bekap(magasság(i)):=idő(i)
  különben ha beki(i)='OFF'
              akkor bekap(magasság(i)):=hamis
                    ha hang(magasság(i))
                     akkor kikap(magasság(i)):=idő(i)
                           hang(magasság(i)):=hamis
                     különben töröl(i)
  különben ha beki(i)='ON'
              akkor hang(magasság(i)):=igaz
                  ha bekap(magasság(i))=idő(i)
                    akkor töröl(i); bekap(magasság(i)):=idő(i)
  Elágazás vége
  i:=i+1
Ciklus vége
```

A `töröl` eljárás az i-edik utasítást törli a programból.

```
töröl(i):
  Ciklus j=i-től db-1-ig
    beki(j):=beki(j+1); magasság(j):=magasság(j+1)
    idő(j):=idő(j+1)
  Ciklus vége
  db:=db-1
Eljárás vége.
```

Értékelési szempontok

A. Átfedő hang esetén az első OFF-ot a második ON elé teszi	3 pont
1 időegységgel korábbra állítva	2 pont
a régi helyéről megszünteti	2 pont
akkor is jó, ha volt közöttük valami más is	1 pont
B. Ugyanazon időpontú ON és OFF sorrendjét felcseréli, ha kell	2 pont
az OFF-ot 1 időegységgel előbbre teszi	2 pont
akkor is cserél, ha volt közöttük más	1 pont
akkor is előbbre viszi, ha volt közöttük más	1 pont
C. A futás végén a megmaradt hangokat kikapcsolja	2 pont
D. Az ON nélküli OFF-okat elhagyja	2 pont
E. Az azonos időpontú ON-ok közül csak egyet hagy	2 pont
F. Minden más parancsot meghagy	3 pont
G. Felismeri a program végét	1 pont
H. Felismeri az állomány végét	1 pont

2. feladat: 80 nap alatt a Föld körül (75 pont)

A feladat egy gráfbejárás szimulálása, illetve egy gráfban legrövidebb út keresés. Első lépésként a PONT(N) tömbbe kell beolvasni a gráf pontjainak nevét a VAROSx.BE állományból. Ezután a JARATx.BE állományból olvassuk be a gráf éleit. Különlegessége a feladatnak, hogy két pont között több él is lehet (különböző jármű). A gazdaságos memóriakezelés érdekében olyan éllistát kell felépíteni, ahol az azonos pontok közötti élek egymás után vannak, s a gráf csúcsmátrixa (i,j) indexű eleme a megfelelő éllistára hivatkozik. Az élekre öt jellemzőt adunk meg (eszköz, első járat, menetidő, várakozás, díj). Az éleket visszafelé is felvesszük, de az első indulás idejét a jóra állítjuk b. Harmadik lépésként az úton (UTITERVx.BE állomány) végighaladva számoljuk az eltelt időt és az elfogyott pénzt. A legvégén használjuk fel a hosszúsági fokokat arra, hogy a keleti irányú haladásnál levonunk, a nyugatinál pedig hozzáadunk 1 napot az eltelt időhöz.

```
Szimuláció:
  i:=Sorszáma('London')[3]; idő:=0; pénz:=80000; táv:=0;
  Ciklus amíg nem vége(UTITERV)
    Olvas(UTITERV,hova,mivel); j:=Sorszáma(hova)
    Kiválasztás(i,j,mivel,első,menetidő,várakozás,ár)
    Ciklus amíg (idő-első) mod (2*menetidő+2*várakozás)≠0
      idő:=idő+1
    Ciklus vége
```

[3] Megadja az adott város sorszámát a Pont tömbben.

```
    pénz:=pénz-ár; táv:=táv+fok(j)-fok(i)
    idő:=idő+menetidő; i:=j
  Ciklus vége
  Ha táv≠0 akkor idő:=idő-táv/360
Eljárás vége.
```

A második részfeladat, az optimális útiterv kidolgozása egy gráfbejárási feladat, melyben olyan kört kell keresni Londonból, amely megkerüli a földet 80 nap alatt és elég rá a 80 000 font. Ezek közül is a legjobbat kell megtalálni, azaz a legolcsóbbak közül a leghamarabb célba érőt. A bejáráshoz használt prioritási sorban a városok sorszáma, az odaérkezési idő, a megtett távolság és az addig elköltött pénzösszeg van. A sorba akkor nem veszünk fel egy várost egy útvonallal, ha van olyan ideérkezés, amely korábbi is és olcsóbb is. Ha viszont ő jobb a sorban levő bármelyiknél (azaz abban a városban korábbi is és olcsóbb is), akkor a sorból a megfelelőket törölni kell.

```
Bejárás:
  i:=Sorszáma('London'); PrSorba(i,0,0,0); Halmazba(i)
  Ciklus amíg nem üres a PrSor
    PrSorból(i,idő,táv,pénz)
    Ciklus j=1-től N-ig
      Ha vanél(i,j) akkor Elsőél(i,j,k); Lépés(i,j,k)
                          Ciklus amíg vankövetkezőél(i,j)
                            Következőél(i,j,k); Lépés(i,j,k)
                          Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.

Lépés(i,j,k):
  Ha nem (j=Sorszáma('London') és táv+fok(j)-fok(i)=0) akkor
    Ha pénz+k.ár≤80000 akkor {van elég pénz rá}
      oda:=odaérés(idő,k); ár:=pénz+k.ár
      Ha oda≤80-előjel(k) akkor {80 napon belül}
        Ha nem vanjobb(j,oda,ár) akkor {nem lehet jobban}
          Ha j=Sorszáma('London') akkor felír(oda-táv/360,ár)
                különben PrSorba(j,oda,táv+fok(j)-fok(i),ár)
Eljárás vége.

Odaérés(idő,k):
  Ha idő<k.első akkor idő:=k.első
  Ciklus amíg (idő-k.első) mod (2*k.menetidő+k.2*várakozás)≠0
    idő:=idő+1
  Ciklus vége
  odaérés:=idő
Függvény vége.
```

Értékelési szempontok

A. 1. rész: kiírja a városokat és az eszközöket	2 pont
jól számolja az eltelt időt	3 pont
jól számolja a szükséges járatra várakozási időt	3 pont
jól számolja a megmaradt pénzt	3 pont
észreveszi, ha elfogy a pénze	2 pont
észreveszi, ha nem utazta körül a földet	6 pont
észreveszi, ha 80 napnál tovább tartott az út	1 pont
észreveszi, ha 80 napnál nem volt hosszabb az út	1 pont
keleti irányú körbejárásnál levon 1 napot	2 pont

nyugati irányú körbejárásnál hozzáad 1 napot	2 pont
B. 2. rész: talál 80 napnál rövidebb utat (ha van)	6 pont
olyat talál, amire elég a pénze (ha van)	5 pont
észreveszi, ha nincs 80 napnál rövidebb,	3 pont
észreveszi, ha nincs elég pénze	3 pont
ezek közül azt választja, ami a legolcsóbb	3 pont
ezek közül azt választja, ami a leggyorsabb	3 pont
keleti irányú körbejárást választ	2 pont

1996. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rajzolgatás (25 pont)

A pontszám akkor is megadható, ha a rajzolt ábráról leolvasható a válasz. *Jel* helyett *szakasz, vonaldarab* stb., *pont* helyett *egy egység vonal, egységnyi hosszúságú vonal* stb. is jó.

A. — — — — — — — — — — — —

n db jelből álló szaggatott vonal	3 pont
5 egység vonal, 5 egység szünet	1+1 pont

B. • • • • • • • • • • • • •

n db jelből álló pontozott vonal	3 pont
1 pont, 5 egység szünet	1+1 pont

C. • — • — • — • — • — • — •

n db jelből álló pontozott-szaggatott vonal	3 pont
1 pont, 5 egység szünet	1+1
5 egység vonal, 5 egység szünet	1+1 pont

D. • — — • — — • — — • — —

n db jelből álló pontozott-szaggatott vonal	3 pont
1 pont, 5 egység szünet	1+1 pont
kétszer: 5 egység vonal, 5 egység szünet	1+1+1 pont

2. feladat: Mit csinál? (24 pont)

A. X értékét írja ki	4 pont
2-es számrendszerben	4 pont
hátulról visszafelé	4 pont
B. 1-et ír ki, ha X bitjei között páratlan számú 1-es van	4 pont
0-t, ha az 1-esek száma páros	4 pont
C. X 1-es értékű bitjeinek számát írja ki	4 pont

3. feladat: Szövegelés (18 pont)

A. 80 karakteresre egészíti ki és	2 pont
jobbra igazítva írja ki	3 pont
B. 80 karakteresre egészíti ki és	2 pont
balra igazítva írja ki	3 pont
C. 80 karakteresre egészíti ki és	2 pont
középre igazítva írja ki	4 pont
Ha páratlan számú karakter volt S-ben, akkor előtte 1-gyel több szóköz lesz, mint mögötte	2 pont

4. feladat: Vacakánykészítő Művek (33 pont)

Egyetlen lehetséges műveletsorrend van:

	Péter	Jakab	Károly
10.00	laposít (kevel)	kevel (laposít)	hegyez
10.30	lapostyát fest		likaszt
11.00	lakkoz		sorszámoz
11.30	szegentyűt fest		
12.00	ebédel	szerel	ebédel
12.30	csomagol	ebédel	iktat

A. Pontosan 13 órakor 3 pont

B. Jakab 12.30 és 13.00 között ebédel 4 pont

C. Péter csomagolja a vacakányt 5 pont

D. A lapostyát festik előbb 6 pont

E. Péter ötöt, Jakab kettőt, Károly négyet 5+5+5 pont

Kilencedik-tizedik osztályosok

1. feladat: Rajzolgatás (18 pont)

A. □ □ □ □ □ □ □ □ □ □ □ □ □ □

n db négyzetből álló sorminta 3 pont

oldalhossza 5 egység, közöttük levő szünet 5 egység 1+1 pont

B. Δ ∇ Δ ∇ Δ ∇ Δ ∇ Δ ∇ Δ ∇ Δ ∇

n db egyenlő oldalú háromszögből álló sorminta 3 pont

minden második háromszög fordított állású 2 pont

oldalhossza 5 egység, közöttük levő szünet 5 egység 1+1 pont

C. ✳ ✳ ✳ ✳ ✳ ✳ ✳ ✳ ✳ ✳ ✳ ✳

n db csillagból álló sorminta 3 pont

mindegyik 4 db 4 egység hosszú szakaszból áll, közöttük 4 egység szünet 1+1+1 pont

2. feladat: Logikai műveletek (12 pont)

A. Ha az `A` vektor minden eleme nagyobb az őt megelőzőnél 4 pont

B. Ha az A vektor legalább egy eleme nagyobb az őt megelőzőnél 4 pont

C. N=2 esetén egyformán működnek 4 pont

3. feladat: Kördiagram (14 pont)

A: A körcikkeket egymásra rajzolja 2 pont

Félkörre normálja a körcikkek szögét az egész kör helyett 2 pont

A körcikkek festéséhez egyetlen színt használ, az `N` kódút 2 pont

B. Egy jó megoldás ciklusmagja:

```
Z:=SZ; SZ:=SZ+A(I)/S*360
Körcikk(Z,SZ,I)
```

180 helyett 360-nal szoroz 1 pont

a szögek összegét számolja 2 pont

jó kezdő- és végszöget használ a körcikknél 2+2 pont

jól választ színt 1 pont

4. feladat: Levelek (19 pont)

```
közvetlenFőnöke(A,F) ha igazgató(F) és igazgatóhelyettes(A) vagy      3 pont
   igazgatóhelyettes(F) és osztályvezető(A,O) vagy                    3 pont
   osztályvezető(F,O) és alkalmazott(A,O).                            3 pont
```

(1) Egy lehetséges nem-rekurzív megoldás:

```
beosztottja(F,A) ha igazgató(F) és nem igazgató(A) vagy               3 pont
   igazgatóhelyettes(F) és nem
   (igazgató(A) vagy igazgatóhelyettes(A)) vagy                       4 pont
```

E két utóbbi sor helyett például a következő kettő is jó:

```
      igazgatóhelyettes(F) és
      (osztályvezető(A,O) vagy alkalmazott(A,O)) vagy
```

```
   osztályvezető(F,O) és alkalmazott(A,O).                            3 pont
```

(2) Egy lehetséges rekurzív megoldás:

```
beosztottja(F,A) ha közvetlenFőnöke(A,F) vagy                         3 pont
   közvetlenFőnöke (F1,F) és beosztottja(F1,A).                       7 pont
```

Megjegyzés: Nem-rekurzív és rekurzív megoldás egyaránt elfogadható, a részfeladat megoldására mindkét esetben ugyanannyi, azaz max. 10 pont adható.

5. feladat: Vasútmodell (18 pont)

i	1	2	3	4	5	6	7	8	9	10	11
X	0	1	4	9	4	1	0				
V	0	2	4	6	4	2	0				
Y	1	3	5	7	5	3	1				
Vúj	2	4	6	-	6	4	2				
megtett út	**0**	1	4	9	11	16	19				

Minden helyesen beírt szám 0,5 pontot ér, kivéve a 11-et és a - jelet, amelyek 1-1 pontot érnek. (A bal alsó sarokba beírt **0**-ért természetesen nem jár pont.) A - helyett 0 is elfogadható az adott cellában.

6. feladat: Csak párhuzamosan! (19 pont)

A. 3 lépés, 2 processzor 3+2 pont

Részpontszámok, ha nem a jó választ adta:

1. lépésben `b*c` és `d*e` számítása 2 processzorral 2 pont
2. lépésben `a+b*c` számítása (`a+d*e` vagy `b*c+d*e` is jó) 2 pont
3. lépésben `a+b*c+d*e` számítása 1 pont

B. 5 lépés, 2 processzor 5+2 pont

Részpontszámok, ha nem a jó választ adta:

1. lépés: `d*e, f/g` (bármelyik helyett a `h*i` is jó) 2 pont
2. lépés: `d*e+f/g`, `h*i` (az 1. lépéstől függően másképp is lehet) 2 pont

3. lépés: `d*e+f/g+h*i`, `a*b`
(`a*b` vagy itt, vagy a 4. lépésben szerepeljen) 1 pont

4. lépés: `c*(d*e+f/g+h*i)` 1 pont

5. lépés: `a*b+c*(d*e+f/g+h*i)` 1 pont

C. 4 lépés, 3 processzor 5+2 pont

Részpontszámok, ha nem a jó választ adta:

1. lépés: `b+c, e/f, g*h` 2 pont

2. lépés: `a*(b+c), i*j, e/f-g*h` 2 pont

3. lépés: `a*(b+c)+i*j, d*(e/f-g*h)` 2 pont

4. lépés: `a*(b+c)-d*(e/f-g*h)+i*j` 1 pont

Megjegyzés: A megoldás 3. lépésében kihasználtuk az összeadás kommutativitását, a feladat szövege azonban nem szól arról, hogy a kifejezés-kiértékelő felismeri-e ezt a tulajdonságot. Ha a versenyző utal arra, hogy a kommutatív tulajdonság figyelembe vétele nélkül a feladat csak 5 lépésben oldható meg 3 processzorral, a teljes pontszám megadható a C részfeladatra, egyébként az 5 lépéses, 3 processzoros megoldásra a 3. lépés részpontszáma (2 pont) nem adható meg.

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Szövegelés (7 pont)

A. Az `S`-ben levő első szót a szöveg elejéről a végére helyezi 3 pont

Ha `S` egyetlen szóból áll (nincs benne szóköz), akkor változatlanul hagyja 1 pont

Ha `S` üres, akkor változatlanul hagyja 1 pont

B. Az első értékadás a szöveg végére ír egy szóközt, amely az utolsó szót elválasztja a mögé írandótól 1 pont

Az utolsó értékadás az eredeti első és második szót elválasztó szóközt hagyja el 1 pont

2. feladat: Mit csinál? (10 pont)

A. `A` = `X` legnagyobb elemének indexe 1 pont

`B` = `X` legkisebb elemének indexe 1 pont

B. `N+N/2-2` 2 pont

Részmegoldás: `N+N/2` 1 pont

C. Akkor, ha `X` összes eleme egyenlő, 2 pont
továbbá `N` = 1 1 pont
vagy `N div` 2^k = 3 (másképpen $3*2^k \le N < 4*2^k$), ahol $k \ge 0$, egész. 3 pont
Ha képlet helyett példákat sorol fel (l. alább), 3 pont helyett csak 2 pont

Magyarázat:

(1) Az algoritmus addig felezgeti a tömböt, amíg két- és egyelemű résztömbökre nem bontja. Kételemű résztömbben `A` és `B` mindig különböző értéket vesz fel, akkor is, ha a két elem egyenlő. Ezért `A` és `B` csak akkor lehet egyenlő, ha a résztömb egyelemű.

(2) A `Ha X(A1) > X(A2) ...`, illetve a `Ha X(B1) < X(B2) ...` választásokban szereplő értékadások közül egyenlőség esetén a `különben` ágban lévőt hajtja végre a program, azaz az elemek egyenlősége esetén a két éppen vizsgált résztömb közül a jobb oldali indexeit kapjuk eredményül `A`-ban, illetve `B`-ben. Nyilvánvaló, hogy ha a teljes tömb összes eleme egyenlő, végül jobb szélső egy- vagy kételemű résztömbjének indexeit kapjuk eredményül.

(3) (1)-ből és (2)-ből következik, hogy A és B csak akkor lehet egyenlő (a triviális N = 1 eseten kívül), ha a teljes tömb összes eleme egyenlő és a felbontása után kapott jobb szélső résztömbje egyelemű.

Az összefüggés kifejezhető zárt képlettel: A akkor és csak akkor lesz egyenlő B-vel, ha N = 1 vagy van olyan egész k, amelyre N div 2^k = 3 (vagy másképpen: $3*2^k \le N < 4*2^k$). Az algoritmus ui. a 3 elemű tömböt 2:1 arányban osztja fel. 3 elemű résztömböt 6 vagy 7 elemű tömb jobb oldalaként kapunk, 6 vagy 7 elemű résztömböt 12, 13, 14 vagy 15 elemű tömb jobb oldalaként kapunk s.í.t.

3. feladat: Mit számol? (10 pont)

A. A = a sokszög területe — 5 pont

B = a sokszög kerülete — 2 pont

B. az óramutató járásával egyező körüljárás esetén korrigálja a negatív értéket — 3 pont

4. feladat: Csak párhuzamosan! (16 pont)

A. 3 lépés, 2 processzor — 2+1 pont

Részpontszámok, ha nem a jó választ adta:

1. lépésben b*c és d*e számítása 2 processzorral — 1 pont
2. lépésben a+b*c számítása (a+d*e vagy b*c+d*e is jó) — 1 pont
3. lépésben a+b*c+d*e számítása — 1 pont

B. 5 lépés, 2 processzor — 4+3 pont

Részpontszámok, ha nem a jó választ adta:

1. lépés: d*e, f/g (bármelyik helyett a h*i is jó) — 2 pont
2. lépés: d*e+f/g, h*i (az 1. lépéstől függően másképp is lehet) — 2 pont
3. lépés: d*e+f/g+h*i, a*b (a*b vagy itt, vagy a 4. lépésben szerepeljen) — 1 pont
4. lépés: c*(d*e+f/g+h*i) — 1 pont
5. lépés: a*b+c*(d*e+f/g+h*i) — 1 pont

C. 4 lépés, 3 processzor — 4+2 pont

Részpontszámok, ha nem a jó választ adta:

1. lépés: b+c, e/f, g*h — 2 pont
2. lépés: a*(b+c), i*j, e/f–g*h — 2 pont
3. lépés: a*(b+c)+i*j, d*(e/f–g*h) — 1 pont
4. lépés: a*(b+c)–d*(e/f–g*h)+i*j — 1 pont

Megjegyzés: A megoldás 3. lépésében kihasználtuk az összeadás kommutativitását, a feladat szövege azonban nem szól arról, hogy a kifejezés-kiértékelő felismeri-e ezt a tulajdonságot. Ha a versenyző utal arra, hogy a kommutatív tulajdonság figyelembe vétele nélkül a feladat csak 5 lépésben oldható meg 3 processzorral, a teljes pontszám megadható a C részfeladatra, egyébként az 5 lépéses, 3 processzoros megoldásra a 3. lépés részpontszáma (2 pont) nem adható meg.

5. feladat: Pascal-program struktúrája (20 pont)

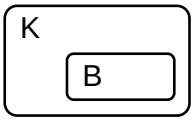
A.
```
Procedure A;                    1 pont
Procedure B;                    1 pont
   Procedure C;                 1 pont
   end {C};
end {B};
```

```
Procedure D;                                                    1 pont
end {D};
end {A};
Procedure E;                                                    1 pont
end {E};
```

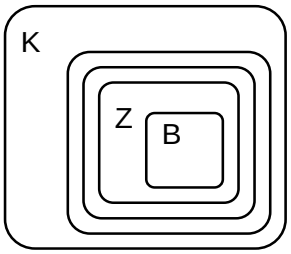
B. Két megoldást is bemutatunk. Az első *két segédeljárást* használ.

```
{B benneVan K-ban}
(1) benneVan(B,K) ha tartalmazza(K,B) vagy                      1 pont
(2) tartalmazza(Z,B) és benneVan(Z,K).                          2 pont
```

az (1) esethez

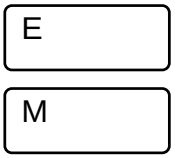


a (2) esethez

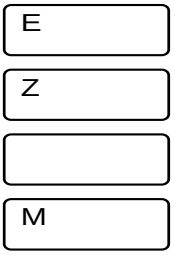



```
{E előbbVanMint M}
(3) előbbVanMint(E,M) ha megelőzi(E,M) vagy                     1 pont
(4) megelőzi(E,Z) és előbbVanMint(Z,M).                         2 pont
```

a (3) esethez

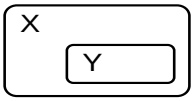


a (4) esethez




```
(5) hívhatja(X,Y) ha tartalmazza(X,Y) vagy                      1 pont
```

az (5) esethez

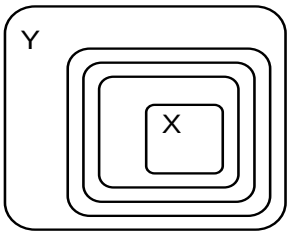



```
(6) X=Y vagy                                                    2 pont
```

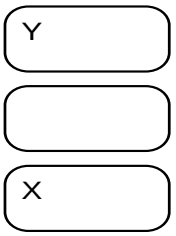
a (6) esethez


```
(7) benneVan(X,Y) vagy                                          1 pont
```

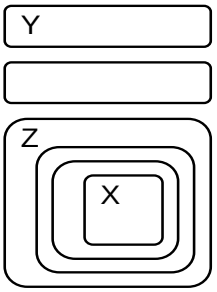
a (7) esethez



(8) `előbbVanMint(Y,X) vagy` 1 pont

a (8) esethez 

(9) `benneVan(X,Z) és előbbVanMint(Y,Z).` 4 pont

a (9) esethez 

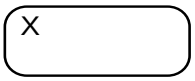
Részmegoldás:

`hívhatja(X,Y) ha tartalmazza(X,Y) vagy`	1 pont
`X=Y vagy`	2 pont
`tartalmazza(Y,X) vagy`	1 pont
`megelőzi(Y,X).`	1 pont

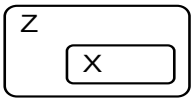
A második valamivel tömörebb, és csak *egy segédeljárásra* van szüksége.

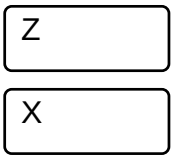
`{X nincsElőbbMint Y}`

(a) `nincsElőbbMint(X,Y) ha X=Y vagy` 1 pont

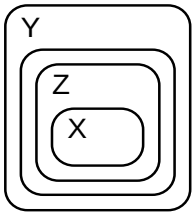
az (a) esethez 

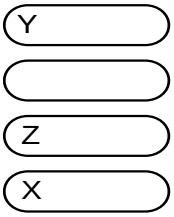
(b) `(tartalmazza(Z,X) vagy megelőzi(Z,X)) és` 4+4 pont

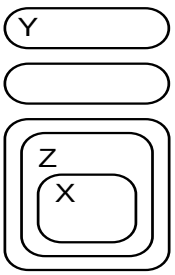
a (b) esethez 

vagy 

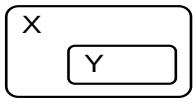
(c) `nincsElőbbMint(Z,Y).` 4 pont

a (c) esethez 

vagy 

vagy 

(d) `hívhatja(X,Y) ha tartalmazza(X,Y) vagy` 1 pont

a (d) esethez 

(e) `nincsElőbbMint(X,Y).` 1 pont

Megjegyzés: A rajzok a javítást segítik, a versenyzőktől programszöveget várunk.

6. feladat: Multihalmazok (24 pont)

A. Egyik(A,B): a két gazdának együtt hány állata van 3 pont
A példa esetén: `221`

Másik(A,B): az egyes állatfajtákból fajtánként hány van 5 pont
A példa esetén:
`[[kacsa,6],[kecske,33],[liba,13],[nyúl,165],[tyúk,4]]`

B. (az itt közölttől eltérő, helyes megoldás is elfogadható)

```
Hányféle(A,B):
Ha üres(A) és üres(B) akkor 0                                   1 pont
különben ha üres(A) akkor 1+Hányféle(A,elsőutániak(B))          1 pont
különben ha üres(B) akkor 1+Hányféle(elsőutániak(A),B))         1 pont
különben ha első(első(A))>első(első(B)) akkor
   1+Hányféle(A,elsőutániak(B))                                 2 pont
különben ha első(első(A))<első(első(B)) akkor
   1+Hányféle(elsőutániak(A),B))                                2 pont
különben 1+Hányféle(elsőutániak(A),elsőutániak(B))              2 pont
Függvény vége.
```

Ha a két gazda állatfajtáinak számát csak külön-külön tudja megadni, akkor 3 pont

C. (az itt közölttől eltérő, helyes megoldás is elfogadható)

```
Eladhatók(A,B):
Ha üres(A) vagy üres(B) akkor []                                1 pont
különben ha első(első(A))>első(első(B)) akkor
   Eladhatók(A,elsőutániak(B))                                  2 pont
különben ha első(első(A))<első(első(B)) akkor
   Eladhatók(elsőutániak(A),B))                                 2 pont
különben Elejére(pár(első(első(A)),
                min(második(első(A)),második(első(B)))),
            Eladhatók(elsőutániak(A),elsőutániak(B)))           2 pont
Függvény vége.
```

7. feladat: Pénzgyűjtögetők (13 pont)

A. Ha egymás után két 10 forintos jött 2 pont

vagy ha két 10 forintos között páros számú 5 forintos jött 3 pont

Részmegoldás: csak a 'két 10-es között két 5-ös' esetet említi 1 pont

B. `Ha PÉNZ=5 akkor PÉNZ:=10; Ezüst Benő` 5 pont

C. Ha 5 forintos következne, az előző 10-essel együtt azonnal kiírnánk, és így ha megint 5 forintos jönne, akkor a két ötöst Arany Oszkár nem tudná 10-esre váltani. 3 pont

1996. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Sportverseny (20 pont)

A feladat olyan X és Y egész számpár megtalálása, amelyre X*5+Y*7 a legjobban megközelíti alulról N-et. A képlet alapján X≤N div 5. Így a megoldás:

```
X:=N div 5; kész:=hamis; jóX:=0; jóY:=0
Ciklus amíg X≥0 és nem kész
  Y:=(N-X*5) div 7
  Ha X*5+Y*7>jóX*5+jóY*7 akkor jóX:=X; jóY:=Y
  X:=X-1
  kész:=(jóX*5+jóY*7=N)
Ciklus vége
Ki: jóX,jóY,N-jóX*5-jóY*7
```

Értékelési szempontok

A. Az osztálylétszám beolvasása 1 pont

B. Ír ki olyan eredményt, amelyben a csapatlétszámok összege nem nagyobb az osztálylétszámnál (pl. 16 → 3,0,1 a helyes megoldás, de a 16 → 2,0,6 vagy a 16 → 1,1,4 válasz is 3 pontot ér; 0 pontos pl. a 16 → 2,1,? válasz. 3 pont

C. Jó eredményt ad 5-tel osztható osztálylétszámra (pl. 15 → 3,0,0) 3 pont

D. Jó eredményt ad 7-tel osztható osztálylétszámra (pl. 21 → 0,3,0) 3 pont

E. Jó eredményt ad 5x+7y típusú osztálylétszámra (pl. 29 → 3,2,0) 5 pont

F. Jó eredményt ad általános esetben (pl. 23 → 3,1,1) 5 pont

EGYÉB:

ha a 0 maradékot nem írja ki, a >0-t igen, akkor megadható a maximális pont

ha maradékot egyáltalán nem ír, akkor a 3 pontos tesztesetek 2, az 5 pontosak pedig 3 pontot érnek

2. feladat: Hét törpe (28 pont)

A megoldásban beolvasás közben kell figyelni, hogy az adott név volt-e már, s a megfelelő számlálóval számolni:

```
Be: NÉV; db:=0
Ciklus amíg NÉV≠''
  i:=1
  Ciklus amíg i≤db és NÉV≠Törpe(i).név
    i:=i+1
  Ha i≤db akkor Törpe(i).db:=Törpe(i).db+1
       különben Törpe(i).db:=1; Törpe(i).név:=NÉV
  Ciklus vége
  Be: NÉV
Ciklus vége
```

Értékelési szempontok

A. Egyetlen név begépelésekor jó választ ad (pl. Kuka →1/Kuka,1) 1+1+1 pont

B. Egy nevet többször gépelünk be, de csak egyszer írja ki, helyes előfordulási számmal (pl. Hapci, Hapci, Hapci →1/Hapci,3) 1+2+2 pont

C. Több nevet gépelünk be, de mindegyiket csak egyszer (pl. Hapci, Kuka, Morgó, Szundi, Szende, Tudor, Vidor → 7/Hapci,1/Kuka,1/Morgó,1/Szundi,1/Szende,1/Tudor,1/Vidor,1) 2+3+3 pont

D. Van egyszer és van többször szereplő név is, és mindegyiket egyszer írja ki (pl. Hapci, Kuka, Tudor, Kuka, Kuka → 3/Hapci,1/Kuka,3/Tudor,1) 2+5+5 pont

3. feladat: Órák angolul (27 pont)

Egy konstans tömbben érdemes tárolni a számokat 1-től 12-ig:

```
Szám=(one,two,three,four,five,six,seven,eight,nine,ten, ele-
ven,twelve)
```

A programnak a beolvasott óra és perc ismeretében döntenie kell, hogy melyik fajta kiírást választja:

```
Ha óra<12 akkor X:='a.m.' különben X:='p.m.'
Ha óra>12 akkor óra:=óra mod 12
Ha perc=0 akkor Ki: Szám(óra),X
különben ha perc=15 akkor Ki: 'a quarter past ',Szám(óra),X
különben ha perc=30 akkor Ki: 'half past ',Szám(óra),X
különben Ki: 'a quarter to ',Szám(óra+1),X
```

Értékelési szempontok

A. A délelőttöt és délutánt jól tudja (pl. 5,0 → five o'clock a.m., 16,0 → four o'clock p.m.) 3+3 pont

B. Az összes egész órát jól tudja 12*1 pont

C. A negyedeket jól tudja (pl. 18,15 → a quarter past six p.m.) 3 pont

D. A feleket jól tudja (pl. 7,30 → half past seven a.m.) 3 pont

E. A háromnegyedeket jól tudja (pl. 8,45 → a quarter to nine a.m.) 3 pont

Kilencedik-tizedik osztályosok

1. feladat: Bűvös négyzet (18 pont)

A feladat megoldásában egy mátrixban a bűvös négyzetben szereplő minden irányra a számok összegének kiszámolása:

```
A:=0; B:=0
Ciklus i=1-től N-ig
  A:=A+Bűvös(i,i); B:=B+Bűvös(i,N-i+1)
Ciklus vége
Jó:=(A=B)
Ha Jó akkor Sorvizsgálat(Jó,A)
Ha Jó akkor Oszlopvizsgálat(Jó,A)
```

A Sorvizsgálat ellenőrzi minden sor összegét, illetve, hogy a számok ugyanabban a sorban mind különbözőek-e! A különbözőség vizsgálatához előállítja a sor elemeit tartalmazó halmazt:

```
Sorvizsgálat(Jó,A):
  i:=1
  Ciklus amíg i≤N és Jó
    h:={}; S:=0
    Ciklus j=1-től N-ig
      S:=S+Bűvös(i,j); h:=h∪{Bűvös(i,j)}
    Ciklus vége
    Jó:=(S=A és h={1..N})
    i:=i+1
  Ciklus vége
Eljárás vége.
```

Az Oszlopvizsgálat ehhez nagyon hasonló:

```
Oszlopvizsgálat(Jó,A):
  j:=1
  Ciklus amíg j≤N és Jó
    h:={}; S:=0
    Ciklus i=1-től N-ig
      S:=S+Bűvös(i,j); h:=h∪{Bűvös(i,j)}
    Ciklus vége
    Jó:=(S=A és h={1..N})
    j:=j+1
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A. Helyes bűvös négyzetre az IGEN választ adja	5 pont
B. Észrevesz rossz sorösszeget	2 pont
C. Észrevesz rossz oszlopösszeget	3 pont
D. Észrevesz rossz átlóösszeget	2 pont
E. Észrevesz soron belüli számismétlődést	2 pont
F. Észrevesz oszlopon belüli számismétlődést	2 pont
G. Egy bűvös négyzetben többféle hibát is észrevesz	2 pont

2. feladat: Névnapok (24 pont)

Legyenek a beolvasott nevek a NAPTÁR(365,5) mátrixban úgy, hogy amelyik napon 5-nél kevesebb névnap van, ott üres érték szerepel! Az A részfeladat megoldásához azon nevek számát kell megadni, amelyek a naptárban korábban még nem szerepeltek:

```
Nevek száma:
  db:=0
  Ciklus i=1-től 365-ig
    Ciklus j=1-től 5-ig
      Ha NAPTÁR(i,j)≠'' akkor
        k:=1; l:=1
        Ciklus amíg k<i és NAPTÁR(i,j)≠Naptár(k,l)
          Ha l=5 vagy NAPTÁR(k,l+1)='' akkor k:=k+1; l:=1
                                        különben l:=l+1
        Ciklus vége
        Ha k=i akkor db:=db+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A B részfeladat ennek egy egyszerűsítése, végig kell nézni a naptárt:

```
Névnapok:
  Ciklus i=1-től 365-ig
    Ciklus j=1-től 5-ig
      Ha NAPTÁR(i,j)=NÉV akkor Ki: i
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A.	Naponta egy név, minden név egyszer	2 pont
	Naponta egy név, vannak többször előforduló nevek	3 pont
	Egyes napokon több név, minden név egyszer	3 pont
	Egyes napokon több név, vannak többször előforduló nevek	4 pont
B.	A név egyszer van a naptárban és egyedül van a sorban	2 pont
	A név többször van a naptárban és egyedül van a sorban	3 pont
	A név nem egyedül van a sorban, de ő az első	3 pont
	A név nem egyedül van a sorban, és nem ő az első	4 pont

EGYÉB:

adható a nevek számára részpont, ha ugyanazt a nevet többször is számolja: az A részfeladat 2. és 4. tesztje ekkor 2-2 pontot ér.

3. feladat: Alakzatok (18 pont)

Az alakzatok egymásba helyezését kell megfogalmaznunk:

- Négyzetbe kör: ha az átmérője kisebb vagy egyenlő, mint a négyzet oldalhossza.
- Négyzetbe háromszög: ha a háromszög oldalhossza kisebb vagy egyenlő, mint a négyzet oldalhossza osztva cos 15°-tel (az egyik csúcsa a négyzet egyik sarkában lesz).
- Körbe négyzet: ha az átlója kisebb vagy egyenlő, mint a kör átmérője.
- Körbe háromszög: ha a háromszög magasságának kétharmada kisebb vagy egyenlő, mint a kör sugara.
- Háromszögbe négyzet: ha a négyzet oldalhossza kisebb vagy egyenlő, mint a háromszög oldalhossza szorozva gyök(3)/(2+gyök(3))-mal.
- Háromszögbe kör: ha a kör sugara kisebb vagy egyenlő, mint a háromszög magasságának harmada.

Ha az alábbi eljárás végén a db változó 0 marad, akkor jó az egymásba helyezés.

```
Ellenőrzés:
  db:=0
  Ciklus i=2-től N-ig
    Ha nem Belefér(i,i-1) akkor db:=db+1; rossz(db):=i
  Ciklus vége
Eljárás vége.
```

```
Belefér(i,j):
  gy:=gyök(3)
  Ha Fajta(i)=Fajta(j) akkor Belefér:=(Méret(i)≤Méret(j))
  különben ha Fajta(i)=2 és Fajta(j)=1
              akkor Belefér:=(Méret(i)*2≤Méret(j))
  különben ha Fajta(i)=3 és Fajta(j)=1
              akkor Belefér:=(Méret(i)*cos(pi/12)≤Méret(j))
  különben ha Fajta(i)=1 és Fajta(j)=2
              akkor Belefér:=(Méret(i)*gyök(2)/2≤Méret(j))
  különben ha Fajta(i)=3 és Fajta(j)=2
              akkor Belefér:=(Méret(i)*gy*2/3≤Méret(j))
  különben ha Fajta(i)=1 és Fajta(j)=3
              akkor Belefér:=(Méret(i)*(2+gy)/gy≤Méret(j))
  különben ha Fajta(i)=2 és Fajta(j)=3
              akkor Belefér:=(Méret(i)≤Méret(j)*gy/6)
Eljárás vége.
```

Értékelési szempontok

A *jó* szó jelentése a pontozáshoz: ha belefér, igent mond, ha nem fér bele, akkor nemet. Azaz: *jó* = belefér ÉS (válasz = IGEN) VAGY NOT(belefér) ÉS (válasz = NEM).

A. Négyzetben négyzet *jó*	1 pont
B. Négyzetben kör *jó*	1 pont
C. Négyzetben háromszög *jó*	3 pont
D. Körben négyzet *jó*	2 pont
E. Körben kör *jó*	1 pont
F. Körben háromszög *jó*	3 pont
G. Háromszögben négyzet *jó*	3 pont
H. Háromszögben kör *jó*	3 pont
I. Háromszögben háromszög *jó*	1 pont

4. feladat: Sportverseny (15 pont)

A feladat olyan X, Y, Z egész számhármas megtalálása, amelyre X*5+Y*7+Z*11 a legjobban megközelíti alulról N-et. A képlet alapján X≤N div 5. Így a megoldás:

```
X:=N div 5; kész:=hamis; jóX:=0; jóY:=0; jóZ:=0
Ciklus amíg X≥0 és nem kész
  Y:=(N-X*5) div 7
  Ciklus amíg Y≥0 és nem kész
    Z:=(N-X*5-Y*7) div 11
    Ha X*5+Y*7+Z*11>jóX*5+jóY*7+jóZ*11
       akkor jóX:=X; jóY:=Y; jóZ:=Z
    Y:=Y-1
    kész:=(jóX*5+jóY*7+jóZ*11=N)
  Ciklus vége
  X:=X-1
Ciklus vége
Ki: jóX,jóY,N-jóX*5-jóY*7
```

Értékelési szempontok

A. Jó eredményt ad 5-tel osztható osztálylétszámra (pl. 15 → 3,0,0,0)	1 pont
B. Jó eredményt ad 7-tel osztható osztálylétszámra (pl. 21 → 0,3,0,0)	1 pont
C. Jó eredményt ad 11-tel osztható osztálylétszámra (pl. 11 → 0,0,1,0)	1 pont

D. Jó eredményt ad 5x+7y típusú osztálylétszámra (pl. 24 → 2,2,0,0) 1 pont

E. Jó eredményt ad 5x+11z típusú osztálylétszámra (pl. 16 → 1,0,1,0) 1 pont

F. Jó eredményt ad 7y+11z típusú osztálylétszámra (pl. 18 → 0,1,1,0) 1 pont

G. Jó eredményt ad 5x+7y+11z típusú osztálylétszámra (pl. 23 → 1,1,1,0) 1 pont

H. Jó eredményt ad általános esetben, ha van maradék (pl. 13 → 1,1,0,1) 1+2 pont

I. Csapatszámra maximalizál (pl. 1996 → 397,0,1,0) 5 pont

EGYÉB:

ha a 0 maradékot nem írja ki, a >0-t igen, akkor megadható a maximális pont

ha maradékot egyáltalán nem ír, akkor a pontszám 2/3-a adható.

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Sportverseny (12 pont)

A feladat szinte megegyezik az előző kategória 4. feladatával, egyetlen különbséggel: nem mindegy, hogy a több egyforma megoldás közül melyiket írjuk ki:

```
X:=N div 5; kész:=hamis; jóX:=0; jóY:=0; jóZ:=0; db:=0; A:=0
Ciklus amíg X≥0 és nem kész
  Y:=(N-X*5) div 7
  Ciklus amíg Y≥0 és nem kész
    Z:=(N-X*5-Y*7) div 11
    B:=X*5+Y*7+Z*11
    Ha B>A akkor jóX:=X; jóY:=Y; jóZ:=Z; A:=B
    különben ha B=A és X+Y+Z>db
                akkor jóX:=X; jóY:=Y; jóZ:=Z; db:=X+Y+Z
    Y:=Y-1
    kész:=(jóX*5+jóY*7+jóZ*11=N)
  Ciklus vége
  X:=X-1
Ciklus vége
Ki: jóX,jóY,N-jóX*5-jóY*7
```

Értékelési szempontok

A. Jó eredményt ad 5-tel osztható osztálylétszámra (pl. 15 → 3,0,0,0) 1 pont

B. Jó eredményt ad 7-tel osztható osztálylétszámra (pl. 21 → 0,3,0,0) 1 pont

C. Jó eredményt ad 11-tel osztható osztálylétszámra (pl. 11 → 0,0,1,0) 1 pont

D. Jó eredményt ad 5x+7y típusú osztálylétszámra (pl. 24 → 2,2,0,0) 1 pont

E. Jó eredményt ad 5x+11z típusú osztálylétszámra (pl. 16 → 1,0,1,0) 1 pont

F. Jó eredményt ad 7y+11z típusú osztálylétszámra (pl. 18 → 0,1,1,0) 1 pont

G. Jó eredményt ad 5x+7y+11z típusú osztálylétszámra (pl. 23 → 1,1,1,0) 1 pont

H. Jó eredményt ad általános esetben, ha van maradék (pl. 13 → 1,1,0,1) 1 pont

I. Csapatszámra maximalizál (pl. 385 → 77,0,0,0) 1 pont

J. Csapatszámra maximalizál nagy létszámra is (pl. 123456789 → 24691355,2,0,0) 3 pont

EGYÉB:

ha a 0 maradékot nem írja ki, a >0-t igen, akkor megadható a maximális pont

ha maradékot egyáltalán nem ír, akkor a pontszám 2/3-a adható.

2. feladat: Kerítésfestés (19 pont)

A feladat megfogalmazható úgy, hogy a H szám előállítható-e az X_1, X_2,.., X_n, közül a lehető legkevesebb összegeként vagy sem! A megoldás dinamikus programozás:

$$Meg(H,i) := \min\begin{cases} Meg(H,i-1) \\ Meg(H-X_i,i-1)+1 \\ 1 \quad ha \quad H=X_i \end{cases}, \quad Meg(H,1) := \begin{cases} 1 \quad ha\ H=X_1 \\ +\infty\ ha\ H \neq X_1 \end{cases}$$

A feladat tehát Meg(H,N) kiszámítása feljegyzéses módszerrel (kezdetben az M tömböt +∞ értékekkel feltöltve):

```
Meg(H,i):
  Ha H=X(i) akkor M(H,i):=1
  különben ha i=1 akkor M(H,i):=+∞
  különben ha M(H,i)=+∞
              akkor M(H,i):=min(Meg(H,i-1),Meg(H-X(i),i-1)+1)
  Meg:=M(H,i)
Függvény vége.
```

A feladat nem oldható meg, ha Meg(H,N) értéke +∞ lesz.

Értékelési szempontok

A. Felismeri, ha nem festhető be, mert nincs elég festék	1 pont
B. Felismeri, ha nem festhető be, mert marad valamelyik festékből	4 pont
C. Tud megoldást adni, ha pont elég a festék a kerítés befestéséhez	1 pont
D. Tud megoldást adni, ha éppen az első K doboz elegendő	2 pont
E. Tud megoldást adni, ha egyetlen megoldás van	5 pont
F. Tud optimális megoldást adni akkor is, ha nem az első az	6 pont

3. feladat: Autóverseny (26 pont)

Értékelési szempontok

Mivel az eredmény valós számokat tartalmaz, helyességüket elég 1 tizedes pontossággal ellenőrizni.

A. Egyenes szakaszon M értékre növeli a sebességet (és nem lépi túl)	2 pont
B. A gyorsítást, illetve lassítást helyesen számítja ki	5 pont
C. Kanyar előtti egyenes szakaszon csak annyit nő a sebessége, hogy a kanyarhoz érve K-ra csökkenhessen	7 pont
D. A kanyarhoz érve éppen K-ra nő (vagy csökken) a sebessége	5 pont
E. Ha a kanyarba K-nál kisebb sebességgel lép be, a kanyarból változatlan sebességgel lép ki	2 pont
F. Ha a kanyarba K sebességgel lép be, akkor K sebességgel is lép ki	2 pont
G. A pálya megtételéhez szükséges időt a három pályára helyesen számolja	1+1+1 pont

4. feladat: Nyomtatás (18 pont)

Értékelési szempontok

A. Jó és jó helyen van az iskola bevezető sora (lapdobás előtte, mögötte)	1+1 pont
B. Jó és jó helyen van az évfolyam bevezető sora (lapdobás előtte, üres sor mögötte)	1+1 pont
C. Jó az osztály bevezető sora	1 pont
D. Jó helyen van az osztály bevezető sora (oldal közepén is, elején is)	1+1 pont

E. Tanulók kért adatait kiírja, pontosan három sorban	1+1 pont
F. Tanulók adatait nem töri szét két oldalra	1 pont
G. Nincs oldal, amelyen tanuló adatai nélkül lenne osztály bevezető sora	1 pont
H. Jól számolja az iskola összefoglaló sorát (mindhárom adatot)	1+1+1 pont
I. Jól számolja az évfolyam összefoglaló sorát	1 pont
J. Jól számolja az osztály összefoglaló sorát	1 pont
K. Az összefoglaló sorok előtt kihagy egy sort	1 pont
L. Ír fejlécet és a láblécbe lapszámot	1 pont

1996. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Tükör-telefonszámok (16 pont)

A telefonszám lehet körzetszámmal együtt tükörszám, lehet a körzetszám önmagában tükörszám és a telefonszám körzetszám nélküli része is lehet tükörszám:

```
Ciklus i=1-től n-ig
  Ha Tükör(tel(i)) akkor Ki: tel(i)
  különben ha hossz(tel(i))=8
             akkor t1:=tel(1..2); t2:=tel(3..8)
                   Ha Tükör(t1) vagy Tükör(t2)
                      akkor Ki: tel(i)
Ciklus vége
```

```
Tükör(t):
  i:=1; n:=hossz(t)+1
  Ciklus amíg i≤n div 2 és t(i)=t(n-i)
    i:=i+1
  Ciklus vége
  Tükör:=(i>n div 2)
Függvény vége.
```

Értékelési szempontok

A. N beolvasása	1 pont
B. N telefonszám beolvasása	2 pont
C. Észreveszi a budapesti tükör-telefonszámokat Nem jár pont, ha nem ilyeneket is kiír	4 pont
D. Észreveszi a vidéki tükör-telefonszámokat Nem jár pont, ha nem ilyeneket is kiír	4 pont
E. Észreveszi a vidéki tükör-körzetszámokat Nem jár pont, ha nem ilyeneket is kiír	1 pont
F. Észreveszi a vidéki körzetszám nélküli tükör-telefonszámokat Nem jár pont, ha nem ilyeneket is kiír	4 pont

2. feladat: Névsor (31 pont)

Mivel a magyar ábécé jeleire a < reláció nem az ábécésorrendű, ezért a szövegekre a kisebb relációt újra kell írni. Szükség van ehhez egy konstans tömbre a betűk jó sorrendjével:

```
abc ='aábcdeéfghiíjklmnoóöőpqrstuúüűvwxyz
      AÁBCDEÉFGHIÍJKLMNOÓÖŐPQRSTUÚÜŰVWXYZ'
```

```
Jósorrend(név1,név2):
  i:=1; h1:=hossz(név1); h2:=hossz(név2)
  Ciklus amíg i≤h1 és i≤h2 és név1(i)=név2(i)
    i:=i+1
  Ciklus vége
  Ha i>h1 akkor Jósorrend:=igaz
  különben ha i>h2 akkor Jósorrend:=hamis
  különben j:=1
           Ciklus amíg j≤hossz(abc) és név1(i)≠abc(j)
             j:=j+1
           Ciklus vége
           k:=1
           Ciklus amíg k≤hossz(abc) és név2(i)≠abc(k))
             k:=k+1
           Ciklus vége
           Jósorrend:=(j<k)
Függvény vége.
```

Nézzük meg hogy hány elem nagyobb az előtte levőnél! Ha 0 vagy n-1, akkor a sorrend csökkenő, illetve növekvő. Egyéb esetben pedig meg kel nézni, hogy milyen relációból van több!

```
Jó:=igaz; növ:=0
Ciklus i=1-től n-1-ig
  Ha Jósorrend(név(i),név(i+1)) akkor növ:=növ+1
                              különben Jó:=hamis
Ciklus vége
Ha növ=n-1 vagy növ=0
   akkor ha Jó akkor Ki: 'Növekvő'
            különben Ki: 'Csökkenő'
különben ha növ≥n div 2 akkor Ki: 'Inkább növekvő'
                      különben Ki: 'Inkább csökkenő'
         Ciklus i=1-től n-1-ig
           Ha Jósorrend(név(i),név(i+1))
              akkor ha növ<n div 2 akkor Ki: név(i),név(i+1)
           különben ha növ≥n div 2 akkor Ki: név(i),név(i+1)
         Ciklus vége
Elágazás vége
```

Értékelési szempontok

A. N beolvasása, N tanuló nevének beolvasása	1+3 pont
B. Észreveszi, ha növekvő sorrendben vannak	3 pont
C. Észreveszi, ha csökkenő sorrendben vannak	3 pont
D. Észreveszi, ha inkább növekvő sorrendben vannak	4 pont
E. Észreveszi, ha inkább, csökkenő sorrendben vannak	4 pont
F. A rossz sorrendben levő párokat kiírja	
az ékezet nélküli betűket jól hasonlítja össze	3 pont
a nagy ékezetes betűket jól hasonlítja össze	2 pont
a kicsi ékezetes betűket jól hasonlítja össze	3 pont
a szóközöket jól hasonlítja össze	2 pont
G. Az összes ilyet kiírja	3 pont

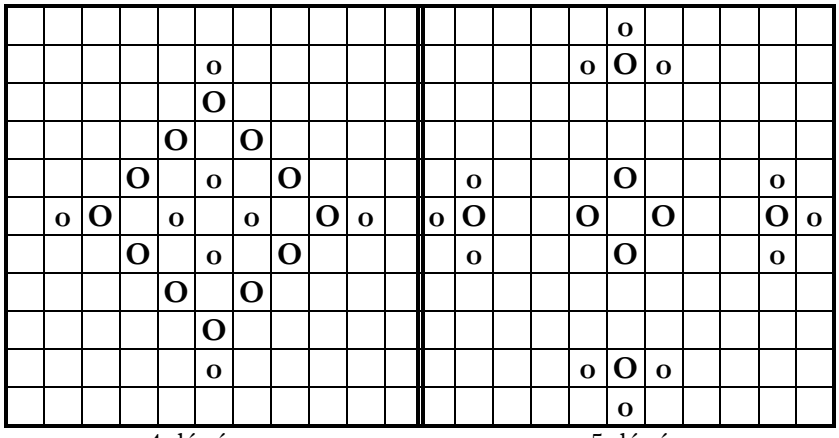
3. feladat: Szimmetriajáték (23 pont)

A feladatbeli 3 szabály a megoldásban 2 esetre csökkenthető.

```
Ciklus amíg szükséges
  segéd:=tábla
  Ciklus i=1-től maxn-ig
    Ciklus j:=1-től maxn-ig
      Ha segéd(i,j)>0 akkor tábla(i,j):=(segéd(i,j)+1) mod 3
      különben ha Szomszédszám(i,j)=1 akkor tábla(i,j):=1
    Ciklus vége
  Ciklus vége
  Táblarajz
Ciklus vége
```

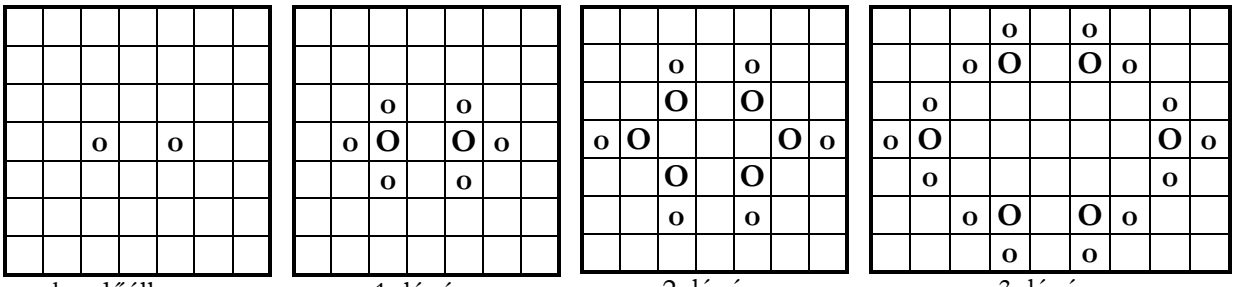
Értékelési szempontok

A feladatszövegbeli példa alapján a pontozás nagyrészt elvégezhető, csupán azt kell ellenőrizni, hogy valóban szimulálja-e a folyamatot vagy csupán a példabeli ábrákat rajzolja ki a képernyőre. Ez az ellenőrzés elvégezhető a példa továbbfuttatásával:



4. lépés 5. lépés

valamint 2 kiinduló pont vizsgálatával:



kezdőállapot 1. lépés 2. lépés 3. lépés

A. Beolvassa a kezdőpontok számát	1 pont
B. Tud egy pontot elhelyezni a kezdőállapotban, beolvassa a koordinátáját	1 pont
C. Tud két pontot elhelyezni a kezdőállapotban, beolvassa a koordinátáit	2 pont
D. Billentyűlenyomásonként lép egyet	1 pont
E. A kis **o** betűk váltanak nagy **O**-ra, de más nem	3+3 pont
F. A nagy **O** betűk eltűnnek, de más nem	2+2 pont
G. A megfelelő üres helyeken jelennek meg az új kis **o** betűk	8 pont

Ha nem minden irányban jelennek meg, akkor irányonként 2-2 pont adható. Ha csak kis **o** szomszédokat vizsgál, akkor 5 pont adható

Kilencedik-tizedik osztályosok

1. feladat: Logo (75 pont)

Érdemes a beolvasást intelligensen megírni, azaz olvassunk a bemenetről szavanként:

```
db:=0; pr(0).kód:=0; fent:=hamis
Ciklus amíg nem vége(f)
  Szóolvasás(f,szó); Utasításkód(szó,kód)
  db:=db+1; pr(db).kód:=kód
  Ha kód≤4 akkor Szóolvasás(f,szó); Számmá(szó,pr(db).par)
Ciklus vége
```

A program optimalizálása veszi az utasításokat, s meghívja a szükséges optimalizáló lépést (egyszerűsítve, csak konstans paraméterekre adjuk meg):

```
i:=1
Ciklus amíg i≤db
  Elágazás pr(i).kód szerint
    6: le(i)
    5: fel(i)
    1: előre(i)
    2: hátra(i)
    3: balra(i)
    4: jobbra(i)
   egyébként i:=i+1
  Elágazás vége
Ciklus vége
```

DOWN utasításra nincs szükség, ha a toll le van engedve, illetve ha őt egy UP előzte meg, akkor egyikükre sincs szükség:

```
le(i):
  Ha nem fent akkor kihagy(i,i)
  különben fent:=hamis
           Ha pr(i-1).kod=5 akkor kihagy(i-1,i); i:=i-1
                            különben i:=i+1
Eljárás vége.
```

UP utasításra nincs szükség, ha a toll fel van emelve, illetve ha őt egy DOWN előzte meg, akkor egyikükre sincs szükség:

```
fel(i):
  Ha fent akkor kihagy(i,i)
  különben fent:=igaz
           Ha pr(i-1).kod=6 akkor kihagy(i-1,i); i:=i-1
                            különben i:=i+1
Eljárás vége.
```

A 0 hosszúságú lépéseket törölni kell! Két szomszédos elmozdulást össze lehet vonni:

```
előre(i):
  Ha pr(i).par=0 akkor Kihagy(i,i)
  különben ha pr(i-1).kód=1 akkor egyformamozgás(i)
  különben ha pr(i-1).kód=2 akkor különbözőmozgás(i)
  különben i:=i+1
Eljárás vége.
```

Az azonos irányú elmozdulások összevonhatók. A különbözők csak akkor, ha a toll fel van emelve; ebben az esetben a 0 lépéshosszú összevonások megszüntethetők:

```
egyformamozgás(i):
  Ha pr(i).par*pr(i-1).par≥0 vagy fent
     akkor pr(i-1).par:=pr(i-1).par+pr(i).par
           Ha pr(i-1).par=0 akkor Kihagy(i-1,i); i:=i-1
                        különben Kihagy(i,i)
  különben i:=i+1
Eljárás vége.

különbözőmozgás(i):
  Ha pr(i).par*pr(i-1).par<0 vagy fent
     akkor pr(i-1).par:=pr(i-1).par-pr(i).par
           Ha pr(i-1).par=0 akkor Kihagy(i-1,i); i:=i-1
                        különben Kihagy(i,i)
  különben i:=i+1
Eljárás vége
```

A hátrafelé lépés eljárása szinte azonos az előrelépéssel:

```
hátra(i):
  Ha pr(i).par=0 akkor Kihagy(i,i)
  különben ha pr(i-1).kód=2 akkor egyformamozgás(i)
  különben ha pr(i-1).kód=1 akkor különbözőmozgás(i)
  különben i:=i+1
Eljárás vége.
```

A 0 szögű forgásokat törölni kell! Két szomszédos elfordulást össze lehet vonni. Forgás az őt megelőző tollfelemeléssel vagy leengedéssel felcserélhető:

```
balra(i):
  Ha pr(i).par=0 akkor Kihagy(i,i)
  különben ha pr(i-1).kód=3 akkor egyformaforgás(i)
  különben ha pr(i-1).kód=4 akkor különbözőforgás(i)
  különben ha pr(i-1).kód<3 akkor i:=i+1
  különben Csere(pr(i-1),pr(i)); i:=i-1
Eljárás vége.

jobbra(i):
  Ha pr(i).par=0 akkor Kihagy(i,i)
  különben ha pr(i-1).kód=4 akkor egyformaforgás(i)
  különben ha pr(i-1).kód=3 akkor különbözőforgás(i)
  különben ha pr(i-1).kód<3 akkor i:=i+1
  különben Csere(pr(i-1),pr(i)); i:=i-1
Eljárás vége.
```

A forgások összevonása egyszerűbb a mozgásokénál, ugyanis nem kell nézni a toll állapotát:

```
egyformaforgás(i):
  pr(i-1).par:=pr(i-1).par+pr(i).par
  Ha pr(i-1).par=0 akkor Kihagy(i-1,i); i:=i-1
                 különben Kihagy(i,i)
Eljárás vége.

különbözőforgás(i):
  pr(i-1).par:=pr(i-1).par-pr(i).par
  Ha pr(i-1).par=0 akkor Kihagy(i-1,i); i:=i-1
                 különben Kihagy(i,i)
Eljárás vége.
```

Értékelési szempontok

A. A felesleges szóközöket, illetve sorvégeket kiszűri	3+2 pont
B. FORWARD x FORWARD y (azonos előjel vagy felemelt toll)	2+2 pont
C. BACK x BACK y (azonos előjel vagy felemelt toll)	2+2 pont
D. FORWARD x BACK y (ellenkező előjel vagy felemelt toll)	2+2 pont
E. BACK x FORWARD y (ellenkező előjel vagy felemelt toll)	2+2 pont
F. LEFT x LEFT y	2 pont
G. RIGHT x RIGHT y	2 pont
H. LEFT x RIGHT y	2 pont
I. RIGHT x LEFT y	2 pont
J. DOWN forgások UP, DOWN UP	2+2 pont
K. UP forgások DOWN, UP DOWN	2+2 pont
L. DOWN ... DOWN, UP ... UP	2+2 pont
M. Az azonos típusú összevonások többszörös alkalmazása	6 pont
N. A kihagyottak miatti többszörös összevonás alkalmazása	6 pont
O. A 0 fordulatok, illetve elmozdulások megszüntetése	3+3 pont
P. Állandók összeadása, kivonása (teszt 1.,5.,9.13., ill. 16. sora)	3+3 pont
Q. Kifejezések összevonása (jó konstans, jó változók, 0 együtthatójú kihagyása)	2+4+4 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Televízióadások (55 pont)

Az A feladat megoldása egy rendezés adósorszám, azon belül pedig kezdési idő szerinti sorrendben:

```
Afeladat:
  Ciklus i=1-től N-1-ig
    min:=i
    Ciklus j=i+1-től N-ig
      Ha T(j).adó<T(min).adó vagy T(j).adó=T(min).adó
                      és T(j).kezd<T(min).kezd akkor min:=j
    Ciklus vége
    Csere(T(i),T(min))
  Ciklus vége
Eljárás vége.
```

A B feladatban az adatokat a műsorok vége, azon belül pedig kezdése szerint kell sorba rendezni, majd mindig az első műsort választani, amit megnézhetünk (mohó stratégia):

```
Bfeladat:
  Rendezb; ut:=1; Ki: T(1)
  Ciklus i=2-től N-ig
    Ha T(i).kezd≥T(ut).vég akkor Ki: T(i); ut:=i
  Ciklus vége
Eljárás vége.
```

```
Rendezb:
  Ciklus i=1-től N-1-ig
    min:=i
    Ciklus j=i+1-től N-ig
      Ha T(j).vég<T(min).vég vagy T(j).vég=T(min).vég
                        és T(j).kezd<T(min).kezd akkor min:=j
    Ciklus vége
    Csere(T(i),T(min))
  Ciklus vége
Eljárás vége.
```

A C feladat szinte azonos a B feladattal, a mohó stratégiát kétszer kell alkalmazni, s a kiírásban a két keletkezett sorozatot össze kell futtatni:

```
Cfeladat:
  Rendezb; dbt:=1; X(dbt):=T(1); T(1).kezd:=-1
  Ciklus i=2-től N-ig
    Ha T(i).kezd≥X(dbt).vég akkor dbt:=dbt+1; X(dbt):=T(i)
                                  T(i).kezd:=-1
  Ciklus vége
  dbv:=0; Y(dbv):=T(1)
  Ciklus i=2-től N-ig
    Ha T(i).kezd≥Y(dbv).vég akkor dbv:=dbv+1; Y(dbv):=T(i)
  Ciklus vége
  Összefuttat(X,dbt,Y,dbv,Z,db)
Eljárás vége.
```

A D feladat a B feladatra hasonlít, meg kell őrizni a legrövidebb kimaradt filmet, majd az első helyen, ahol megnézhető, be kell venni a megoldásba:

```
Dfeladat:
  Rendezb; vég:=T(1).vég; db:=1; x(db):=1; h:=+∞; ind:=0
  Ciklus i=2-től N-ig
    Ha T(i).kezd-vég≥h és ind>0
                       akkor db:=db+1; x(db):=ind; vég:=vég+h
    Ha T(i).kezd≥vég akkor db:=db+1; x(db):=i; vég:=T(i).vég
    különben ha h=+∞ akkor Vizsgálat(i)
             különben ha h>T(i).vég-T(i).kezd
                         akkor h:=T(i).vég-T(i).kezd; ind:=i
  Ciklus vége
Eljárás vége.

Vizsgálat(i):
  Ha T(x(db)).vég-T(x(db)).kezd<T(i).vég-T(i).kezd és
                         db>1 és T(i).kezd≥T(x(db-1)).vég
     akkor j:=i+1
           Ciklus amíg j≤N és T(j).kezd<vég
             j:=j+1
           Ciklus vége
           Ha j≤ és T(j).kezd≥T(i).vég
              akkor h:=T(x(db)).vég-T(x(db)).kezd
                     ind:=x(db); x(db):=i
   különben h:=T(i).vég-T(i).kezd; ind:=i
Eljárás vége.
```

Értékelési szempontok

A.	Sorbarendezés jó TV-csatorna szerint	3 pont
	Sorbarendezés jó kezdőidő szerint	3 pont
B.	Ha csak egy TV-csatorna van, akkor az összes adást megadja	2 pont

	Ha függetlenek az intervallumok, akkor mind megadja	2 pont
	Ha átfedőek, akkor kiválasztja a maximális számú független	4 pont
	Ha több maximális van, akkor megadja a leghosszabbat	5 pont
	Bonyolult adathalmazra is ad 1 percen belül megoldást	6 pont
C.	Ha nem kell semmit videóra venni, akkor mindent megnézhetünk	3 pont
	Ha csak egyszeres átfedések vannak, akkor maximális számú független választ és minden mást videóra vesz	5 pont 5 pont
	Ha van többszörös átfedés is, akkor a nézettek és a videóra vettek száma maximális	5 pont
D.	Ha csak egyszeres átfedések vannak, s azok befférnek a lyukakba, akkor mindent felvesz és a lyukakban lejátssza őket	2 pont 2 pont
	Nem vesz fel videóra, ha még nem néztük meg a felvettet	4 pont
	A maximális számút nézhetjük meg a felvettekkel együtt	4 pont

2. feladat: Maga mindent kétszer mond, kétszer mond? (20 pont)

Olvassuk be a szabályokat egyesével, s csak azokat vegyük fel a szabálygyűjteménybe, amelyek a korábbiaktól függetlenek! Egy szabálycsoport feldolgozása az alábbi:

```
Szabályok:
  db:=N
  Ciklus I=1-től N-ig
    Szabályolvasás(Szab(i).bal,Szab(i).jobb)
  Ciklus vége
  Ciklus amíg van új szabály
    Ha Független(Új) akkor db:=db+1; szab(db):=Új
                   különben feles:=igaz; Kiírás(Új)
  Ciklus vége
  Ha nem feles akkor Ki: 'Semmi sem felesleges'
Eljárás vége.
```

Az új szabály független a többitől, ha a jobboldala, esetleg több szabály alkalmazása után levezethető a baloldalból:

```
Független(Új)
  le:={Új.bal}
  Ciklus
    i:=1
    Ciklus amíg i≤n és nem
                   (szab(i).bal⊆le és nem(szab(i).jobb⊆le)
       i:=i+1
    Ciklus vége
    bővült:=i≤n
    Ha bővült akkor le:=le∪{szab[i,2]}
  amíg bővült és nem Új.jobb⊆le
  Ciklus vége
  Független:=nem(Új.jobb⊆le)
Függvény vége
```

Értékelési szempontok

A. Felismeri, hogy hány szabálycsoport van	2 pont
B. Szabálycsoporton belül minden szabályt megnéz	1 pont
C. $S->X_i$ típusú szabálycsoporthoz újat elfogad, feleslegeset felismer	1+1 pont
D. $X_i->X_j$ típusú szabálycsoporthoz újat elfogad, feleslegeset felismer	1+1 pont

E. X_i−>X_j...X_k típusú szabálycsoporthoz újat elfogad, feleslegeset felismer	3 pont
F. X_i...X_k−>X_j típusú szabálycsoporthoz újat elfogad, feleslegeset felismer	5 pont
G. A−>B alakú felesleges szabályt felismer	1 pont
H. A−>BCD alakú felesleges szabályt felismer	2 pont
I. ABC−>D alakú felesleges szabályt felismer	3 pont

1997. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendszámok (20 pont)

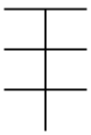
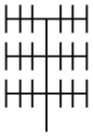
```
Rendszámválasztás:
  REND(1):=Véletlenbetű("G")                          5 pont
  Ciklus I=2-től 3-ig                                 5 pont
     REND(I):=Véletlenbetű("Z")
  Ciklus vége
  REND(4):="-" 5 pont
  Ciklus I=5-től 7-ig                                 5 pont
     REND(I):=Véletlenszámjegy(9)
  Ciklus vége
Eljárás vége.
```

Jó részmegoldás az is, ha 7 független értékadást ír:

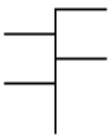
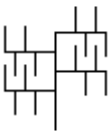
```
REND(1):=Véletlenbetű("G")
REND(2):=Véletlenbetű("Z")
...
```

értékadásonként 2-2 pont

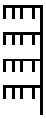
2. feladat: Rajzoló (42 pont)

A.1. 2.  3. 

Helyes rajzonként 4+6+4 pont

B.1. 2.  3. 

Helyes rajzonként 4+6+4 pont

C. 1. 2.  3. 

Helyes rajzonként 4+6+4 pont

3. feladat: Távolugrók (18 pont)

A. Az A a győztes versenyző eredményét fogja tartalmazni 5 pont
a B pedig a második helyezettét 5 pont

Ha az élen holtverseny volt, akkor A és B ugyanazt az értéket tartalmazza 4 pont

B. Ha N<2, azaz csak 1 versenyző indult a versenyen 2 pont

C. Ciklus I=3-tól ... 2 pont

4. feladat: Kerítésmagasságok (20 pont)

A. Ha van olyan lakos, akinek a kerítése magasabb mindkét szomszédjáénál 6 pont

B. A program nem hasonlítja össze az 1., ill. az N. kerítés magasságát a szomszédos kerítések magasságával 2 pont

A két javított sor: 8 pont

```
A:=X(N-1); B:=X(N); L:=hamis
Ciklus I=1-től N-ig
```

Bonyolultabb megoldás, részpontszámért:

```
Ha X(1)>X(2) és X(1)>X(N) akkor L:=igaz
```
3 pont

```
Ha X(N)>X(N-1) és X(N)>X(1) akkor L:=igaz
```
3 pont

C. A `B` az éppen vizsgált, (I–1) sorszámú kerítés magassága 2 pont

Az `A` az azt megelőző, (I–2) sorszámú kerítés magassága 2 pont

Kilencedik-tizedik osztályosok

1. feladat: Halmazműveletek (10 pont)

A. A két halmaz unióját teszi a H halmazba. 3 pont

Az A és a B halmaz üres lesz. 1-1 pont

B. H=A lesz 2 pont

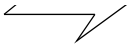
B-ben az eredeti B-ből azok az elemek maradnak, amelyek nem elemei A-nak (B–A) 3 pont

2. feladat: Rajzolgatunk (16 pont)

A. Nyíl 4 pont

a nyílhegy egy egyenlő oldalú háromszög 1 pont

szögei 60 fokosak 1 pont



B. Nyíl 4 pont

a nyíl hátsó végén a szögek 45 fokosak 1 pont

a nyíl eleje egy egyenlő oldalú háromszög 2 oldala 2 pont

a visszahajló szárak szögei 300 fokosak 1 pont

a nyílhegy szárain levő kicsi szögek 30 fokosak 1 pont

nyíl hegyénél a szög 60 fokos 1 pont

3. feladat: Programstruktúrák kezdete és vége (18 pont)

HIBA1: IF nélküli FI 3 pont

HIBA2: IF belsejében lezáratlan DO utasítás van (IF ... DO ... IF) 3 pont

HIBA3: IF belsejében DO nélküli OD van (IF ... OD ... FI) 3 pont

HIBA4: DO nélküli OD 3 pont

HIBA5: lezáratlan IF 3 pont

HIBA6: lezáratlan DO 3 pont

4. feladat: Hőmérsékletmérések (14 pont)

A. Megkeresi a legmelegebbet azok közül a napok közül, amelyeken fagyott (azaz negatív volt a hőmérséklet) — 4 pont
4 pont

Ha több egyforma érték is van, közülük a legkisebb sorszámút adja meg — 2 pont

B. `L`: igaz, ha talált negatív hőmérsékletet — 1 pont

`P`: a megfelelő nap sorszáma, ha L igaz, — 1 pont

`M`: a legnagyobb negatív érték, ha L igaz — 1 pont

`P` és `M`: nemdefiniált, ha `L` hamis — 1 pont

5. feladat: Festőalgoritmus (18 pont)

A. Ha a kép bal széléről indulva páratlan számú határponton haladt át, akkor az alakzat belsejében van, — 3 pont

ha pedig páros számún, akkor kívül. — 3 pont

B. Ha páratlan számú pontból álló vízszintes szakasz esetén mindkét oldalon azonos irányban folytatódik a határvonal (egy ilyen esetet mutat az ábra) — 6 pont

Ha páros számú pontból álló vízszintes szakasz esetén a két oldalon különböző irányban folytatódik a határvonal (egy ilyen esetet mutat az ábra) — 6 pont

Részpontok:

Ha valamelyik sorban az alakzat egy csúcsa, előtte és mögötte pedig külső pont vagy a képernyő széle van; — 2 pont
ha valamelyik sorban az alakzat egy csúcsa, előtte és mögötte pedig belső pont van — 2 pont

6. feladat: Pincérek és vendégek (24 pont)

A. A hiba az, hogy I és J értéke 0 és N–1 között változik, a REND tömböt pedig 1-től N-ig kell indexelni. — 2 pont

Javítása: I:=(I+1) mod N helyett I:=(I mod N)+1, — 2 pont

J:=(J+1) mod N helyett J:=(J mod N)+1 írandó — 2 pont

Más megoldás:

REND(I):=(ÉTEL,VENDÉG) helyett REND(I+1):=(ÉTEL,VENDÉG), — 2 pont

REND(J):=(ÉTEL,VENDÉG) helyett REND(J+1):=(ÉTEL,VENDÉG) kell — 2 pont

B. I:=1; J:=1; DB:=0 — 3 pont

(Ha az A. kérdésre a második megoldást adta, akkor I=0, J=0 a jó megoldás)

C. a 6. percben érkező a 13. percben — 1 pont

a 7. percben érkező a 14. percben — 1 pont

a 9. percben érkező a 18. percben — 1 pont

a 15. percben érkező a 22. percben 2 pont

a 20. percben érkező a 27. percben 2 pont

D. a 10. perctől a 14. 4 pont
és a 17. perctől a 18. percig dolgoznak mindketten 4 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Rekurzió (20 pont)

A. Minden olyan szót (külön-külön sorban), amely a HAPCI szó betűiből előállítható. (Egy betű többször is előfordulhat.) 5 pont

B. Az aktuális szó i. helyén a HAPCI szó hányadik betűje szerepel. 3 pont

C. 5^5-szer (3125) 3 pont

D. Minden olyan szót (külön-külön sorban), amely a HAPCI szó betűiből előállítható. (Egy betű csak egyszer fordulhat elő.) 6 pont

E. 5!-szor (120) 3 pont

2. feladat: Közös adatbázis (18 pont)

A. Egyszerre csak egy író program futhat, 2 pont

és csak akkor, ha egyetlen olvasó program sem dolgozik. 2 pont

Egy időben több program is olvashat. 2 pont

B. Ha egy olvasó program működése közben újabb olvasó program jelez olvasási szándékot, azonnal lehetőséget kap az olvasásra. Ha az olvasási igények olvasás közben folyamatosan jelentkeznek, az író programoknak várakozniuk kell. 3 pont

C. Az olvasó programban a `Vár amíg` sort kell kijavítani: 2 pont
`Vár amíg Van("Írás", "Írási szándék")` 3 pont

D. Ha egy író program írási szándékot jelez, és közben egy másik író programtól újabb írási szándék érkezik, mielőtt az első az írás üzenetet elküldené, akkor vele egyidejűleg a második is elkezdheti az írást. 4 pont

3. feladat: Telefonszámok (12 pont)

A. 9999 1 pont

B. 6624 3 pont

C. Egy olyan számot A és F között, 2 pont

amely nincs benne a TEL vektorban 4 pont

D. $\log_2(8999) \approx 13$ vagy 14 (mindkettő elfogadható) 2 pont

4. feladat: Halmazok (18 pont)

A. Első: D a három halmaz metszete. 4 pont

Második: D elemei azok, amelyek a három halmaz közül pontosan egyben, 3 pont
vagy mind a háromban szerepelnek 3 pont

B. A minimális lépésszám max(N,M,P) 2 pont

A maximális lépésszám N+M+P 2 pont

C. Külső elágazás: A(I), B(J) és C(K) közül legalább kettő egyforma 2 pont

Belső elágazás: A(I), B(J) és C(K) közül mindhárom egyforma 2 pont

5. feladat: Képfeldolgozás (12 pont)

B mátrix: balról jobbra haladva az emelkedő felületek lesznek fényesek, a sík felületek átlagosak, a lejtők pedig sötétek (mintha a felületet nyugati irányból világítanánk meg) 4 pont

C mátrix: balról jobbra és felülről lefelé haladva az inkább emelkedő felületek lesznek fényesek, a sík felületek átlagosak, az inkább lejtő felületek pedig sötétek (mintha a felületet északnyugati irányból világítanánk meg) 4 pont

D mátrix: azok a pontok lesznek az átlagosnál fényesebbek, amelyek a 4 szomszédjuk átlagánál fényesebbek (domború felület), azok lesznek sötétebbek, amelyek a szomszédaik átlagánál sötétebbek (homorú felület), a többiek pedig átlagos fényességűek lesznek. 4 pont

6. feladat: Adatbázis-halmazok (20 pont)

A. Mozi.X: hányan látták az egyes mozikban Rendes Ödön rendező filmjeit 2 pont

Y: az előző nézőszámokból a legnagyobb 2 pont

Mozi.Z: azok a mozik, ahol a legtöbben látták Rendes Ödön rendező filmjeit 2 pont

B. Műsor.X: az egyes műsorokat hányan látták 2 pont

Mozi.Y: az egyes mozik átlagos nézőszáma 2 pont

Film.Z: filmenként azon műsorok (azaz mozik) száma, ahol a filmet többen nézték, mint az illető mozi átlagos nézőszáma 2 pont

C. Film.Vetít=VAN Műsor [Mozi.Város=Budapest] 2 pont

Színész.Keresett=MINDEN Szerep [Film.Vetít] 2 pont

D. Film.Japán=VAN Szerep [Színész.Nemzet="japán"] 2 pont

Mozi.Nincs=NINCS Műsor [Film.Japán] 2 pont

1997. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Határátkelő (24 pont)

Folyamatosan kövessük, hogy az Óriásországból `x(i)` időpontban érkező autós, `p` perc vámvizsgálat után milyen `y(i)` időben lép be Törpeországba.

```
t:=x(1)
Ciklus i=1-től n-ig
  t.perc:=t.perc+p
  Ciklus amíg t.perc≥60
    t.óra:=t.óra+1; t.perc:=t.perc-60
  Ciklus vége
  y(i):=t
  Ha t.óra<x(i+1).óra vagy
                      t.óra=x(i+1).óra és t.perc<x(i+1).perc
     akkor t:=x(i+1)
Ciklus vége
```

Értékelési szempontok

A. Beolvassa P-t és az érkezési időket 1+3 pont

B. Tud órát váltani (példa 2. sorára jól eredményt ad) 5 pont

C. Tud várakozás nélkül átmenni (példa 1. sorára jól eredményt ad) 5 pont

D. Tud várni a sorára (példa 3. sorára jól eredményt ad) 10 pont

2. feladat: Mértékegységek (26 pont)

A megoldáshoz tárolnunk kell a mértékegységek angol és magyar nevét, valamint az angol váltószámokat:

```
amert=('league','mile','fur-
long','pole','yard','fot','inch','line')
mmert=('km','m','dm','cm','mm')
valt=(3,8,40,5.5,3,12,10,2.54)
```

Az angolról magyarra váltás egy képlet kiszámítással, majd pedig osztogatással megvalósítható:

```
t:=0
Ciklus i=1-től n-ig
   t:=(t+x(i))*valt(i)
Ciklus vége
y(1):=egészrész(t/1000000); t:=t-y(1)*1000000
y(2):=egészrész(t/1000); t:=t-y(2)*1000
y(3):=egészrész(t/100); t:=t-y(3)*100
y(4):=egészrész(t/10); y(5):=t-y(4)*10
```

A magyarról angolra váltás is hasonló:

```
t:=x(1)*1000000.0+x(2)*1000+x(3)*100+x(4)*10+x(5)
t:=t/valt(8)
Ciklus i=7-től 1-ig -1-esével
  y(i+1):=t-egészrész(t/valt(i))*valt(i)
  t:=egészrész(t/valt(i))
Ciklus vége
y(1):=t
```

Értékelési szempontok

A. Angol → magyar (1, 1, 1, 1, 1, 1, 1,1 → 6 km 644 m 8 dm 2 cm 0.34 mm)	16 pont
Ha nem jó, akkor kipróbálandók az alábbiak:	
1 league = 4 km 828 m 0 dm 3 cm 2 mm = 4828032 mm	2 pont
1 mile = 1 km 609 m 3 dm 4 cm 4 mm = 1609344 mm	2 pont
1 furlong = 201 m 1 dm 6 cm 8 mm = 201168 mm	2 pont
1 pole = 5 m 0 dm 2 cm 9.2 mm = 5029.2 mm	2 pont
1 yard = 9 dm 1 cm 4.4 mm = 914.4 mm	2 pont
1 foot = 3 dm 0 cm 4.8 mm = 304.8 mm	2 pont
1 inch = 2 cm 5.4 mm = 25.4 mm	2 pont
1 line = 2.54 mm	2 pont
B. Magyar → angol (1,1,1,1,1 → 4 furlong 38 pole 5 yard 2 foot 5 inch 8.189 line)	10 pont
Ha nem jó, akkor kipróbálandók az alábbiak:	
1 km = 4 furlong 38 pole 4 yard 1 foot 10 inch 0.787 line	2 pont
1 m = 1 yard 0 foot 3 inch 3.7 line	2 pont
1 dm = 3 inch 9.37 line	2 pont
1 cm = 3.937 line	2 pont
1 mm = 0.3937 vonás	2 pont

3. feladat: Karácsonyfa (25 pont)

Meg kell számolni, hogy melyik irányban hány szaloncukor van, s a fa akkor dől el, ha valamelyik irányban 25%-kal több szaloncukor van, mint az ellenkező irányban:

```
észak:=0; dél:=0; kelet:=0; nyugat:=0
Ciklus i=1-től n-ig
  Elágazás x(i) szerint
    'e': észak:=észak+1
    'd': dél:=dél+1
    'k': kelet:=kelet+1
    'n': nyugat:=nyugat+1
  Elágazás vége
Ciklus vége
északra:=(észak≥1.25*dél és észak>0)
délre:=(dél≥1.25*észak és dél>0)
keletre:=(kelet≥1.25*nyugat és kelet>0)
nyugatra:=(nyugat≥1.25*kelet és nyugat>0)
```

Értékelési szempontok

A. Nem dől el, ha a négy irányban egyforma mennyiség van (E D K N)	2 pont
B. Nem dől el kis különbség esetén sem (E E E E E E D D D D D)	3 pont
C. Tud dőlni mind a 4 főégtájba (E E D, E D D, K N K, N N K)	2-2-2-2 pont
D. Tud dőlni a mellékégtájakba (E D E K, E D N E, D K, D N D E)	3-3-3-3 pont

Kilencedik-tizedik osztályosok

1. feladat: Üvegválogatás (15 pont)

Először rendezzük sorba a rekeszeket a fehér üvegek száma szerint növekvő sorrendbe:

```
Ciklus i:=2-től n-ig
  j:=i-1; x:=rekesz(i)
  Ciklus amíg j>0 és rekesz(j)>x
    rekesz(j+1):=rekesz(j); j:=j-1
  Ciklus vége
  rekesz(j+1):=x
Ciklus vége
```

Ezután induljunk el elölről, illetve hátulról! Ha az első rekeszben kevesebb a fehér, mint az utolsóban a színes, akkor azokat mind elcseréljük. Majd lépünk a következőre. Ha az utolsóban kevesebb a színes, mint az elsőben a fehér, akkor azokat cseréljük, s lépünk az előzőre. Ha egyforma a darabszámuk, akkor mind elcseréljük, s mindkét irányban lépünk:

```
e:=1; u:=n; csere:=0
Ciklus amíg e<u
  Ha rekesz(e)<m-rekesz[u]
     akkor csere:=csere+rekesz(e)
           rekesz(u):=rekesz(u)+rekesz(e); e:=e+1
  különben ha rekesz(e)>m-rekesz(u)
     akkor csere:=csere+m-rekesz(u)
           rekesz(e):=rekesz(e)-(m-rekesz(u)); u:=u-1
  különben csere:=csere+rekesz(e); e:=e+1; u:=u-1
Ciklus vége
```

Értékelési szempontok

A. Észreveszi, ha csak 1 rekesz van	2 pont
B. Tud egyetlen üveget cserélni	2 pont
C. Észreveszi, ha minden rekesz egyszínű üveget tartalmaz	3 pont
D. Észreveszi, ha csak egyetlen vegyes rekesz van	3 pont
E. Jól cserél általános esetben	5 pont

2. feladat: Benzinkút (21 pont)

A feladat hasonló az 5-8. osztályos határátkelős feladathoz, csak itt nem egyetlen, hanem K helyen lehet várakozni. Az `x(i)` időben érkező autós a legelső szabad `th(j)` töltőhelyre megy, majd a tankolás után `y(i)`-kor távozik:

```
Ciklus i=1-től n-ig
  j:=legkisebb(th)
  Ha th(j).óra<x(i).óra vagy
     th(j).óra=x(i).óra és th(j).perc<x(i).perc
         akkor th(j):=x(i)
  th(j).perc:=th(j).perc+tank(i)
  Ciklus amíg th(j).perc≥60
    th(j).óra:=th(j).óra+1; th(j).perc:=th(j).perc-60
  Ciklus vége
  y(i):=th(j)
Ciklus vége
```

```
Legkisebb:
  min:=1
  Ciklus j=2-től k-ig
    Ha th(j).óra<th(min).óra vagy
       th(j).óra=th(min).óra és th(j).perc<th(min).perc
           akkor min:=j
  Ciklus vége
  legkisebb:=min
Függvény vége.
```

Értékelési szempontok

A. Órákat, perceket jól váltja	4 pont
B. 1 töltőhely, nem kell várni	3 pont
C. 1 töltőhely, kell várni	3 pont
D. 2 töltőhely, nem kell várni	3 pont
E. 3 töltőhely, kell várni (jó és jó sorrendben)	2+2 pont

F. 3 töltőhely, kell várni (jó és jó sorrendben) 2+2 pont

3. feladat: Vonatok (21 pont)

Megadjuk a kukutyini és a piripócsi indulás idejét a két vonathoz (x és y tömbök), az állomások távolságát (táv) és várakozási idejét (vár):

```
vár:=0; x(0):=kukutyin; y(n+1):=piripócs; i:=0; j:=n+1
Ciklus amíg i≤n és j≥1
  Ha kisebb(x(i),y(j)) akkor
          i:=i+1; x(i):=x(i-1); növel(x(i),táv(i))
          Ha nem kisebb(x(i),y(j)) és i=j
             akkor kukutyini:=hamis; hely:=i
                   vár:=60*(x(i).óra-y(j).óra)+
                        x(i).perc-y(j).perc; y(j):=x(i)
          Elágazás vége
          növel(x(i),áll(i))
  különben j:=j-1; y(j):=y(j+1); növel(y(j),táv(j+1))
          Ha nem kisebb(y(j),x(i)) és i=j
             akkor kukutyini:=igaz; hely:=i
                   vár:=60*(y(j).óra-x(i).óra)+
                        (y(j).perc-x(i).perc); x(i):=y(j)
          növel(y(j),áll(j))
  Ha i=j akkor hely:=i
Ciklus vége
Ciklus amíg i≤n
  i:=i+1; x(i):=x(i-1); növel(x(i),táv(i)+áll(i))
Ciklus vége
Ciklus amíg j≥1
  j:=j-1; y(j):=y(j+1); növel(y(j),táv(j+1)+áll(j))
Ciklus vége
```

Pici nehézséget okoz az idő típus kezelése (óra, perc), ezért eljárásokat írunk rá:

```
növel(t,perc):
  t.perc:=t.perc+perc
  Ciklus amíg t.perc≥60
    t.óra:=t.óra+1; t.perc:=t.perc-60
  Ciklus vége
Eljárás vége.
```

```
kisebb(t1,t2):
  kisebb:=t1.óra<t2.óra) vagy
          t1.óra=t2.óra és t1.perc≤t2.perc
Függvény vége.
```

Értékelési szempontok

A. Piripócson találkoznak 2 pont

B. Kukutyinban találkoznak 2 pont

C. Egyszerre érkeznek a 2. állomásra 3 pont

D. A piripócsi indulása előtt megérkezik a kukutyini 3 pont

E. A piripócsi várakozik 4 pont

F. A kukutyini várakozik 4 pont

G. Az órákat-perceket jól váltja 3 pont

4. feladat: Foltkód (18 pont)

Első lépésként adjuk meg a lehetséges szomszéd pontok koordináta-eltéréseit, valamint iránykódjait:

```
ir=((-1,0,2),(-1,-1,4),(0,-1,4),(1,-1,6),(1,0,6),(1,1,8),
(0,1,8),(-1,1,2))
```

Keressük meg az első fekete pontot! Ha van ilyen, akkor nézzük meg, hogy egyetlen pontból áll-e a folt! Ha nem, akkor indulhat a körüljárás:

```
Elsőpontkeresés(k,l,van); ydb:=0
Ha van és nem Egypont(k,l)
   akkor irány:=8; i:=k; j:=l
         Ciklus
           Ciklus amíg x(i+ir(irány,1),j+ir(irány,2))='.'
             Ha irany>1 akkor irány:=irány-1
                        különben irány:=8
             i:=i+ir(irány,1); j:=j+ir(irány,2)
             ydb:=ydb+1; y(ydb):=irány; irány:=ir(irány,3)
         amíg nem(i=k és j=l)
         Ciklus vége
Elágazás vége
```

Az első pont megkereséséhez be kell járnunk a képet fentről lefelé, azon belül pedig balról jobbra:

```
Elsőpontkeresés(i,j,van);:
  i:=1; j:=1; van:=(x(i,j)='X')
  Ciklus amíg i≤n és nem van
    Ha j=m akkor i:=i+1; j:=1 különben j:=j+1
    Ha i≤n akkor van:=(x(i,j)='X')
  Ciklus vége
  van:=(i≤n)
Eljárás vége.
```

Egyetlen pontból áll a folt, ha a szomszédságában nincs más pont:

```
Egypont(k,l):
  db:=0
  Ciklus i=k-1-től k+1-ig Ha i≥1 és i≤n akkor
    Ciklus j=l-1-től l+1-ig Ha j≥1 és j≤m akkor
      Ha x(i,j)='X' akkor db:=db+1
    Ciklus vége
  Ciklus vége
  Egypont:=(db=1)
Függvény vége.
```

Értékelési szempontok

A. Észreveszi, ha nincs folt	2 pont
B. Észreveszi, ha egyetlen pontból áll a folt	2 pont
C. Tud csak főirányba lépni	2 pont
D. Tud csak mellékirányba lépni	2 pont
E. Tud általános foltot	10 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Sereg (16 pont)

Tudjuk, hogy ha egy ember belép a hadseregbe, akkor a főnöke már biztosan ott van, azaz az új katona főnökei száma a közvetlen főnöke főnökei számánál pontosan eggyel több; akinek egyúttal eggyel nő a közvetlen alárendeltjei száma.

```
Ciklus év=1-től h-ig
  Eredményszámítás(év); kiírás(év)
Ciklus vége
```

Az eredményeket évente számoljuk, minden évben csak az addig belépőkkel foglalkozunk.

```
Eredményszámítás(év):
  i:=1
  Ciklus amíg i≤n és ember(i).év≤év
    j:=Kiválaszt(ember(i).főnök)
    ember(i).főszám:=ember(j).főszám+1
    ember(j).bedb:=ember(j).bedb+1
    i:=i+1
  Ciklus vége
```

A zsold az aktuális létszám 10-zel szorozva, s ehhez hozzáadva mindenkire a főnökei számát (akik miatta kapnak 1-1 aranyat).

```
  zsold:=i*10
  Ciklus j=1-től i-1-ig
    zsold:=zsold+ember(j).főszám
  Ciklus vége
```

A B feladat megoldása a 0 beosztottúak száma:

```
  nincs:=0
  Ciklus j=0-tól i-1-ig
    Ha ember(j).bedb=0 akkor nincs:=nincs+1
  Ciklus vége
```

A C feladatnál a maximális beosztott számot határozzuk meg:

```
  max:=0
  Ciklus j=1-től i-1-ig
    Ha ember(j).bedb>ember(max).bedb akkor max:=j
  Ciklus vége
  legtöbb:=ember(max).név
```

A D feladatban a legtöbb főnökszámú ember kapja meg utoljára a zsoldját:

```
  max:=0
  Ciklus j=1-től i-1-ig
    Ha ember(j).főszám>ember(max).főszám akkor max:=j
  Ciklus vége
  vége:=ember(max).főszám+1
Eljárás vége.
```

Értékelési szempontok

A. 1 személyes hadsereg	1 pont
B. Mindenki a tábornok közvetlen beosztottja	2 pont
C. Mindenkinek egy főnöke van	2 pont
D. Általános szerkezet	2+2+2+2 pont
E. Év váltás is jó	3 pont

2. feladat: Járda (15 pont)

Legyen `Járda(i,j)` igaz értékű, ha az i-edik oszlop j-edik sorában törött lap van! Először vizsgáljuk meg azt, hogy az adott oszlop mindkét lapja törött-e! Ezután nézzük meg, hogy valamelyik sor két szomszéd lapját kell-e cserélni! Végül azt nézzük meg, hogy az adott oszlopban van-e egy cserélendő lap!

```
csere:=0
Ciklus i=1-től n-ig
  Ha járda(i,1) és járda(i,2) akkor csere:=csere+1
  különben ha járda(i,1) és járda(i+1,1)
             akkor csere:=csere+1; járda(i+1,1):=hamis
  különben ha járda(i,2) és járda(i+1,2)
             akkor csere:=csere+1; járda(i+1,2):=hamis
  különben ha járda(i,1) vagy járda(i,2) akkor csere:=csere+1
Ciklus vége
```

Értékelési szempontok

A. Egyetlen lapot tud cserélni	1 pont
B. Felismeri az egymás mellett levő lapokat	1 pont
C. Felismeri az egymás alatt levő lapokat	1 pont
D. Felismer több egymás alatt, illetve mellett levő lapot	3 pont
E. A vízszintes irányú csere fontosabb	3 pont
F. Tudja az összes lapot cserélni	3 pont
G. Jól számol szélsőséges helyzetben is	3 pont

3. feladat: Turista (19 pont)

A feladat egy speciális gráfban (térkép) egy speciális tulajdonságú legrövidebb út keresése. Ehhez a szélességi gráfbejárás módosított változatát használjuk, mindig abból a pontból lépünk tovább, ahova az eddigi lépésekkel a leghamarabb értünk oda:

```
táv(1,1):=0; lép(1,1):=0; i:=1; j:=1
Sorinicializálás; Sorba(i,j)
Ciklus amíg (i≠n vagy j≠m) és SOR nem üres
  Sorból(i,j)
  Ha i>1 akkor t:=táv(i,j)+1+|h(i,j)-h(i-1,j)|
              Ha |h(i,j)-h(i-1,j)|≤K
                 akkor Ha táv(i-1,j)>t
                          akkor táv(i-1,j):=t
                                lép(i-1,j):=lép(i,j)+1
                                Sorba(i-1,j)
                       Elágazás vége
              Elágazás vége
  Elágazás vége
  Ha j>1 akkor t:=táv(i,j)+1+|h(i,j)-h(i,j-1)|
              Ha |h(i,j)-h(i,j-1)|≤K
                 akkor Ha táv(i,j-1)>t
                          akkor táv(i,j-1):=t
                                lép(i,j-1):=lép(i,j)+1
                                Sorba(i,j-1)
                       Elágazás vége
              Elágazás vége
  Elágazás vége
```

```
  Ha i<N akkor t:=táv(i,j)+1+|h(i,j)-h(i+1,j)|
               Ha |h(i,j)-h(i+1,j)|≤K
                  akkor Ha táv(i+1,j)>t
                           akkor táv(i+1,j):=t
                                 lép(i+1,j):=lép(i,j)+1
                                 Sorba(i+1,j)
                        Elágazás vége
               Elágazás vége
  Elágazás vége
  Ha j<M akkor t:=táv(i,j)+1+|h(i,j)-h(i,j+1)|
               Ha |h(i,j)-h(i,j+1)|≤K
                  akkor Ha táv(i,j+1)>t
                           akkor táv(i,j+1):=t
                                 lép(i,j+1):=lép(i,j)+1
                                 Sorba(i,j+1)
                        Elágazás vége
               Elágazás vége
  Elágazás vége
Ciklus vége
```

A gráf bejárása után `táv(n,m)`, illetve `lép(n,m)` lesz a megoldás.

Értékelési szempontok

A. N=1, M=1 esetén helyben marad	1 pont
B. N=1, M>1 esetén jobbra megy	1 pont
C. Vízszintes úton halad	3 pont
D. Kis emelkedőn halad	3 pont
E. Kanyargósan halad	5 pont
F. Szerpentinen megy fel	4 pont
G. Nem tud feljutni	2 pont

4. feladat: F.I.L.E (16 pont)

Az I és az L betűnek két olyan pontja van, amely csak egyetlen másik ponttal érintkezik (végpont). Őket úgy lehet megkülönböztetni, hogy az L betűben van olyan pont, ahol derékszögben fordulunk. Az E és az F betűben három végpont van, a különbség: az E-ben kettő, míg az F-ben egy derékszög található. A file-ban szereplő többi betű jellemzői különböznek a fentiektől, azaz ők felismerhetők.

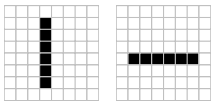
```
Betűfelismerés:
  Számolás(db,sz)
  Ha db=2 akkor Ha sz=0 akkor Betűfelismerés:='I'
                különben ha sz=1 akkor Betűfelismerés:='L'
                különben Betűfelismerés:='-'
  különben ha db=3
           akkor Ha sz=1 akkor Betűfelismerés:='F'
                 különben ha sz=2 akkor Betűfelismerés:='E'
                 különben Betűfelismerés:='-'
  különben Betűfelismerés:='-'
Függvény vége.
```

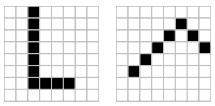
A szomszédok, illetve a derékszögek számolását egy eljárásban végezhetjük:

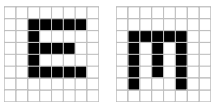
```
Számolás(db,sz):
  db:=0; sz:=0
  Ciklus i=1-től 8-ig
    Ciklus j=2-től 9-ig
      sdb:=-1
      Ha x(i,j)='X'
         akkor Ciklus k=i-1-től i+1-ig
                 Ciklus l=j-1-től j+1-ig
                   Ha x(k,l)='X' akkor sdb:=sdb+1
                 Ciklus vége
               Ciklus vége
               Ha sdb=1 akkor db:=db+1
               különben ha sdb=2 és Derékszög(i,j)
                           akkor sz:=sz+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

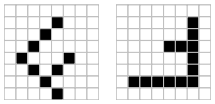
```
Derékszög(i,j):
  Derékszög:=x(i-1,j)='X' és x(i,j+1)='X' vagy
             x(i-1,j-1)='X' és x(i-1,j+1)='X' vagy
             x(i,j-1)='X' és x(i-1,j)='X' vagy
             x(i+1,j-1)='X' és x(i-1,j-1)='X' vagy
             x(i+1,j)='X' és x(i,j-1)='X' vagy
             x(i+1,j+1)='X' és x(i+1,j-1)='X' vagy
             x(i,j+1)='X' és x(i+1,j)='X' vagy
             x(i-1,j+1)='X' és x(i+1,j+1)='X'
Függvény vége.
```

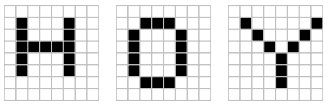
Értékelési szempontok

A.  II 1+2 pont

B.  LL 1+2 pont

C.  EE 1+2 pont

D.  FF 2+2 pont

E.  --- 1+1+1 pont

1997. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Állatkert (25 pont)

Az állatfajok számához meg kell számolni, hogy hány állat neve különbözik a szomszédjától, s ennél eggyel nagyobb a fajok száma:

```
db:=1
Ciklus i=1-től N-1-ig
  Ha x(i)≠x(i+1) akkor db:=db+1
Ciklus vége
```

Az egyes állatok nevét és darabszámát akkor tudjuk meghatározni, amikor a következő fajú állat jön (az utolsó után beteszünk egy fiktív nevet), s közben a maximális darabszámú fajt is figyelhetjük:

```
név:=x(1); db:=1; x(n+1):='###'; maxnév:=név; maxdb:=1
Ciklus i=2-től N+1-ig
  Ha név=x(i) akkor db:=db+1
           különben Ki: név,db
                    Ha db>maxdb akkor maxnév:=név; maxdb:=db
                    név:=x(i); db:=1
Ciklus vége
```

Értékelési szempontok

A. Jó a darabszám, ha egyféle állat van	2 pont
B. Jó az állat neve és száma, ha egyféle állat van	2 pont
C. Jó a legnagyobb létszámú faj, ha egyféle állat van	2 pont
D. Jó a darabszám, ha többféle állat van	5 pont
E. Jó az állatok neve és száma, ha többféle állat van	5 pont
F. Jó a legnagyobb létszámú faj, ha többféle állat van	5 pont
G. Jó az állatok neve és száma, ha többféle állat van, de mindből csak egyetlen	4 pont

2. feladat: Bank (20 pont)

A megoldásban arra kell figyelnünk, hogy a kamatot csak az egy éves lekötés lejártakor szabad hozzáadni a pénzünkhöz, azaz 1 évre visszamenőleg meg kell jegyeznünk, hogy mennyi pénz után kapunk kamatot (kezdetben a `pénz` tömb elemei 0 értékűek):

```
szumma:=0
Ciklus i=1-től N-ig
  Ciklus j=1-től 12-ig
     szumma:=szumma+pénz(j)*x/100+a; Ki: szumma
     pénz(j):=pénz(j)*(1+x/100)+a
  Ciklus vége
Ciklus vége
```

Értékelési szempontok

A. Egy évre jól működik a program (a pénz A, 2*A, 3*A, ..., 12*A)	5 pont
B. Jól számolja a kamatot (a 13. hónapban megjelenik az A*X/100 forint kamat)	5 pont
C. A további években is beteszi az A forintot	5 pont
D. A harmadik évtől már kamatos kamatot számol	5 pont

3. feladat: Balaton (30 pont)

Az A feladat megoldása egy egyszerű megszámolás:

```
napszám:=0
Ciklus i=1-től N-ig
  Ha x(i)>22 akkor napszám:=napszám+1
Ciklus vége
```

A B feladathoz meg kell nézni, hogy mely 7 egymást követő napon volt meleg:

```
időszak:=1
Ciklus amíg időszak≤N-6 és nem melegebb(időszak)
  időszak:=időszak+1
Ciklus vége
Volt:=időszak≤N-6

melegebb(i):
  j:=1
  Ciklus amíg j≤7 és x(i+j-1)>22
    j:=j+1
  Ciklus vége
  melegebb:=(j>7)
Függvény vége.
```

A C feladatnál a leghosszabb meleg szakaszt kell kiválasztani:

```
leghosszabb:=1; hossz:=0; i:=1
Ciklus amíg i≤N
  j:=i
  Ciklus amíg j≤N és x(j)>22
    j:=j+1
  Ciklus vége
  Ha j-i>hossz akkor hossz:=j-i; leghosszabb:=i
  i:=j+1
Ciklus vége
```

Értékelési szempontok

A. A részfeladat megoldása jó	5 pont
B. B részfeladat megoldása jó, ha nincs megoldás	3 pont
C. B részfeladat megoldás jó, ha van pontosan 7 napos időszak	4 pont
D. B részfeladat megoldása jó, ha csak 7 naposnál hosszabb időszak van	6 pont
E. C részfeladat megoldása jó, ha nincs megoldás	3 pont
F. C részfeladat megoldása jó, ha csak 7 naposnál rövidebb időszak van	4 pont
G. C részfeladat megoldása jó, ha csak 7 naposnál hosszabb időszak van	5 pont

Kilencedik-tizedik osztályosok

Feladat: Dél-Afrikai törzsek (75 pont)

Az A részfeladathoz (békés törzsek) meg kell adni azon törzsek számát, akik nem szerepelnek az ellenségeskedők között:

```
Afeladat:
  adb:=0
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg j≤M és ell(j).nép1≠törzs(i) és
                       ell(j).nép2≠törzs(i)
      j:=j+1
    Ciklus vége
    Ha j>M akkor adb:=adb+1; a(adb):=törzs(i)
  Ciklus vége
Eljárás vége.
```

A B részfeladat egy maximumkiválasztás az ellenségeskedés intervallumára:

```
Bfeladat:
  max:=1
  Ciklus i=2-től M-ig
    Ha ell(i).vég-ell(i).kezd>ell(max).vég-ell(max).kezd
       akkor max:=i
  Ciklus vége
  b1:=ell(max).nép1; b2:=ell(max).nép2
Eljárás vége.
```

A C rész megoldását csak abban az esetben hívjuk meg, ha legalább egy háborúskodás volt (M≥1), különben a megoldása a teljes vizsgált időszak:

```
Cfeladat:
  ckezd:=kezdet; cvég:=ell(1).kezd; k:=ell(1).vég
  Ciklus i=2-től M-ig
    Ha ell(i).kezd>k
       akkor v:=ell(i).kezd
             Ha v-k>cvég-ckezd akkor ckezd:=k; cvég:=v
             k:=ell(i).vég
    különben ha ell(i).vég>k akkor k:=ell(i).vég
  Ciklus vége
  Ha vég-k>cvég-ckezd akkor ckezd:=k; cvég:=vég
Eljárás vége.
```

Ha nincs háború (M=0), akkor bármely év (pl. a vizsgált időszak kezdőéve) kiírható, egyébként pedig egy maximumkiválasztást kell végeznünk az éppen ellenségeskedők számára:

```
Dfeladat:
  maxdb:=0
  Ciklus i=kezdet-től vég-ig
    Ciklus j=1-től N-ig
      háborús(j):=hamis
    Ciklus vége
    Ciklus j=1-től M-ig
      Ha i≥ell(j).kezd és i≤ell(j).vég)
         akkor háborús(kiv(ell(j).nép1)):=igaz
               háborús(kiv(ell(j).nép2)):=igaz
    Ciklus vége
```

```
    db:=0
    Ciklus j=1-től N-ig
      Ha háborús(j) akkor db:=db+1
    Ciklus vége
    Ha db>maxdb akkor d:=i; maxdb:=db
  Ciklus vége
Eljárás vége.

Kiv(név):
  i:=1
  Ciklus amíg törzs(i)≠név
    i:=i+1
  Ciklus vége
  kiv:=i
Függvény vége.
```

Az E feladat szinte azonos a D feladattal, csak nem azt kell tárolni, hogy ki volt ellenségeskedő az adott időpontban, hanem azt, hogy hány néppel volt ilyen viszonyban (ezt is csak akkor érdemes meghívni, ha van ellenségeskedés):

```
Efeladat:
  maxdb:=0
  Ciklus i=kezdet-től vég-ig
    Ciklus j=1-től N-ig
      háború(j):=0
    Ciklus vége
    Ciklus j=1-től M-ig
      Ha i≥ell(j).kezd és i≤ell(j).vég)
         akkor s:=kiv(ell(j).nép1); háború(s):=háború(s)+1
               s:=kiv(ell(j).nép2); háború(s):=háború(s)+1
    Ciklus vége
    max:=1
    Ciklus j=2-től N-ig
      Ha háború(j)>háború(max) akkor max:=j
    Ciklus vége
    Ha max>maxdb akkor e:=törzs(max); maxdb:=db
  Ciklus vége
Eljárás vége.
```

Ha a törzseket egy gráf pontjainak, az ellenségeskedéseket pedig egy gráf éleinek fogjuk fel, akkor az F feladat a legnagyobb teljes (bárhonnan bárhova megy él) részgráf meghatározása. Mivel a korlátok kicsik, így a visszalépéses keresés célra vezethet:

```
Ffeladat:
  fdb:=0; x:=0; i:=1
  Ciklus amíg i≥1 és i≤N
    Jóesetkeresés(i,j,van)
    Ha van akkor x(i):=j
                 Ha i>fdb akkor fdb:=i
                                Ciklus j=1-től to fdb-ig
                                  f(j):=törzs(x(j))
                                Ciklus vége
                 Elágazás vége
                 i:=i+1
         különben x(i):=0; i:=i-1
  Ciklus vége
Eljárás vége.
```

```
Jóesetkeresés(i,j,van);:
  j:=x(i)+1; Ha i>1 és j≤x(i-1) akkor j:=x(i-1)+1
  Ciklus amíg j≤N és rossz(i,j)
    j:=j+1
  Ciklus vége
  van:=(j≤n)
Eljárás vége.

rossz(i,j):
  k:=1
  Ciklus amíg k<i és ell(k,j)
    k:=k+1
  Ciklus vége
  rossz:=(k<i)
Függvény vége.

ell(k,j):
  l:=1
  Ciklus amíg l≤M és nem (ell(l).nép1=törzs(x(k)) és
         ell(l).nép2=törzs(j)) vagy ell(l).nép2=törzs(x(k))
         és ell(l).nép1=törzs(j))
    l:=l+1
  Ciklus vége
  ell:=(l≤m)
Függvény vége.
```

Ha a törzseket egy gráf pontjainak, az ellenségeskedéseket pedig egy gráf éleinek fogjuk fel, akkor a G feladat a legnagyobb összefüggő részgráf meghatározása. A törzsek halmazát osszuk fel N darab 1-1 elemű részhalmazra! Vegyük sorra az ellenségeskedéseket, s ha a két benne szereplő törzs külön részhalmazban van, akkor vonjuk össze őket!

```
Gfeladat:
  Ciklus i=1-től N-ig
    halmaz(i):=i; db(i):=1
  Ciklus vége
  Ciklus j=1-től to M-ig
    k1:=kiv(ell(j).nép1); k2:=kiv(ell(j).nép2)
    Ha halmaz(k1)≠halmaz(k2) akkor Összevon(k1,k2)
  Ciklus vége
  max:=1
  Ciklus i=2-től N-ig
    Ha db(i)>db(max) akkor max:=i
  Ciklus vége
  gdb:=0
  Ciklus i=1-től N-ig
    Ha halmaz(i)=max akkor gdb:=gdb+1; g(gdb):=törzs(i)
  Ciklus vége
Eljárás vége.

Összevon(k1,k2):
  y:=halmaz(k2)
  Ciklus i=1-től N-ig
    Ha halmaz(i)=y akkor halmaz(i):=halmaz(k1)
  Ciklus vége
  db(k1):=db(k1)+db(k2); db(K2):=0
Eljárás vége.
```

Értékelési szempontok

A.	Felismer egyáltalán nem háborúzó törzset	5 pont
	Felismeri az összeset	5 pont
B.	Felismeri a leghosszabb ideig ellenségeskedőket	5 pont
C.	Felismeri a leghosszabb békés időszakot	5 pont
	Felismeri a leghosszabb békés időszakot a vizsgált intervallum elején	3 pont
	Felismeri a leghosszabb békés időszakot a vizsgált intervallum végén	3 pont
D.	Felismeri azt az időpontot amikor a legtöbb törzs háborúzott	5 pont
	Akkor is jó az eredmény, ha egy törzs több ellenséggel is háborúzik egyszerre	7 pont
E.	Felismeri azt a törzset, amelyik egyszerre a legtöbb ellenséggel háborúzott	7 pont
F.	Tud kételemű halmazt adni eredményül	5 pont
	Tud legnagyobb teljes részgráfot adni eredményül	10 pont
G.	Tud kételemű halmazt adni eredményül	5 pont
	Tud legnagyobb összefüggő részgráfot adni eredményül	10 pont

Tizenegyedik-tizenharmadik osztályosok

Feladat: Bandák (66 pont)

A feladatok egy gráffal kapcsolatosak, amelynek pontjai a bűnözők (N pont van), él pedig azon pontok között van, akik követtek el együtt bűncselekményt (M él van).

Magányos bűnöző az, aki senkivel nem követett el bűncselekményt (A részfeladat).

```
Afeladat:
  adb:=0
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg j≤M és pár(j,1)≠i és pár(j,2)≠i
      j:=j+1
    Ciklus vége
    Ha j>M akkor adb:=adb+1; a(adb):=i
  Ciklus vége
Eljárás vége.
```

A B részfeladat arról szól, hogy egy gráf hány összefüggő részből áll. Ehhez alkalmazunk egy gráf-bejárást. Praktikus azonban, ha ebben a részben előkészítjük a többi részfeladat megoldását is, azaz megadjuk, hogy mely bűnözők mely bandákban vannak.

```
Bfeladat:
  bandadb:=(0,...,0); volt:=(hamis,...,hamis); bdb:=0
  Ciklus i=1-től N-ig
    Ha nem volt(i) akkor bdb:=bdb+1; bejár(i,bdb)
                         Ha bandadb(bdb)=1 akkor bdb:=bdb-1
  Ciklus vége
Eljárás vége.
```

```
bejár(i,bdb):
  bandadb(bdb):=1; bandák(bdb,bandadb(bdb)):=i
  Sorinicializálás; Sorba(i); volt(i):=igaz
  Ciklus amíg nem üres a SOR
    Sorból(j)
    Ciklus k=1-től M-ig
      Ha pár(k,1)=j és nem volt(pár(k,2))
         akkor bandadb(bdb):=bandadb(bdb)+1
               bandák(bdb,bandadb(bdb)):=pár(k,2)
               sorba(pár(k,2)); volt(pár(k,2)):=igaz
      különben ha pár(k,2)=j és nem volt(pár(k,1))
         akkor bandadb(bdb):=bandadb(bdb)+1
               bandák(bdb,bandadb(bdb)):=pár(k,1)
               sorba(pár(k,1)); volt(pár(k,1)):=igaz
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A C részfeladattal csak akkor foglalkozunk, ha van legalább egy banda. Első lépésként számoljuk le minden bűnözőre, hogy hány kapcsolata van, majd az egyes bandákra (a B részfeladatban meghatározott `bandák` tömb) határozzuk meg ezen számok maximumát!

```
Cfeladat:
  db:=(0,...,0)
  Ciklus i=1-től M-ig
    db(pár(i,1)):=db(pár(i,1))+1
    db(pár(i,2)):=db(pár(i,2))+1
  Ciklus vége
  cdb:=bdb
  Ciklus i=1-től bdb-ig
    max:=bandák(i,1)
    Ciklus j=2-től bandadb(i)-ig
      Ha db(bandák(i,j))>db(max) akkor max:=bandák(i,j)
    Ciklus vége
    c(i):=max
  Ciklus vége
Eljárás vége.
```

A D részfeladatban a B részben kiszámolt `bandadb` tömb maximumát kell megadni:

```
Dfeladat:
  Ha bdb>0 akkor ddb:=bandadb(1)
                 Ciklus i=2-től bdb-ig
                   Ha bandadb(i)>ddb akkor ddb:=bandadb(i)
                 Ciklus vége
       különben ddb:=0
Eljárás vége.
```

Az E részfeladatban minden banda minden tagjára vizsgáljuk meg, hogy a banda hány részre esne, ha az adott tagot kihagynánk, s vegyük bandánként ezen számok maximumát! Ezt is csak akkor hívjuk meg, ha van legalább 1 banda.

```
Efeladat:
  edb:=bdb
  Ciklus i=1-től bdb-ig
    e(i):=bandák(i,1); maxdb:=részek(i,bandák(i,1))
    Ciklus j=2-től bandadb(i)-ig
      db:=részek(bandák(i,j))
      Ha db>maxdb akkor maxdb:=db; e(i):=bandák(i,j)
    Ciklus vége
    f(i):=maxdb
  Ciklus vége
Eljárás vége.
```

A `részek` függvény tulajdonképpen a B részfeladat megismétlése, kis átalakítással:

```
részek(i,index):
  volt:=(igaz,...,igaz); db:=0
  Ciklus k=1-től bandadb(i)-ig
    volt(bandak(i,k)):=hamis
  Ciklus vége
  volt(index):=igaz
  Ciklus k=1-től N-ig
    Ha nem volt(k) akkor db:=db+1; bejár(i)
  Ciklus vége
  részek:=db
Eljárás vége.
```

Az F részfeladat megoldása azon legalább 3 tagú bandák száma, amelyre az E részfeladatban meghatározott `f` tömb 1 elemű.

```
Ffeladat:
  fdb:=0
  Ciklus i=1-től bdb-ig
    Ha f(i)=1 és bandadb(i)>2 akkor fdb:=fdb+1
  Ciklus vége
Eljárás vége.
```

A G részfeladatban is az E részfeladatban meghatározott `f` tömböt használjuk: csak a főnök ismer mindenkit, ha van olyan tag (a főnök), akinek letartóztatása esetén a banda a létszáma-1 részre esik szét.

```
Gfeladat:
  gdb:=0
  Ciklus i=1-től bdb-ig
    Ha f(i)=bandadb(i)-1 akkor gdb:=gdb+1
  Ciklus vége
Eljárás vége.
```

A H részfeladatban minden banda minden tagjára meg kell vizsgálni, hogy ismeri-e az összes többit! Mivel az ismeretség kölcsönös, ezért csak mindenkire csak a nála nagyobb sorszámúakat kell nézni!

```
Hfeladat:
  hdb:=0
  Ciklus i=1-től bdb-ig
    j:=1
    Ciklus amíg j≤bandadb(i) és mindismeri(j,i)
      j:=j+1
    Ciklus vége
    Ha j>bandadb(i) akkor hdb:=hdb+1
  Ciklus vége
Eljárás vége.

mindismeri(i,j):
  k:=j+1
  Ciklus amíg k≤bandadb(i) és ismeri(bandák(i,j),bandák(i,k)
    k:=k+1
  Ciklus vége
  mindismeri:=(k>bandadb(i))
Függvény vége.

ismeri(a,b):
  k:=1
  Ciklus amíg k≤M és nem (pár(k,1)=a és pár(k,2)=b vagy
                          pár(k,1)=b és pár(k,2)=a)
    k:=k+1
  Ciklus vége
  ismeri:=(k≤M)
Függvény vége.
```

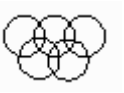
Értékelési szempontok

A.	Megadja a magányos bűnözők sorszámát	3 pont
	Az összes többi sor üres vagy 0, ha csak magányos bűnözők vannak	4 pont
B.	Megadja a bandák számát	5 pont
C.	Megadja az egyes bandák legtöbb kapcsolattal rendelkező tagját	5 pont
D.	Megadja a legnagyobb banda tagjainak számát	5 pont
E.	Megadja az egyes bandák kulcsembereit (ha azok épp a legtöbb kapcsolatúak)	4 pont
	Akkor is megadja, ha azok nem a legtöbb kapcsolatúak	6 pont
F.	Kéttagú bandát nem ad	2 pont
	Megadja a 2 főnél nagyobb biztonságosan szervezett bandák számát	6 pont
G.	Megadja a kéttagú bandák számát	2 pont
	Megadja a 2 főnél nagyobb kockázatosan szervezett bandák számát	6 pont
H.	Megadja a kéttagú bandák számát	2 pont
	Megadja a 2 főnél nagyobb totálisan szervezett bandák számát	6 pont

1998. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Karikák (25 pont)

A.  B.  C.  D.  E. 

Helyes ábránként 5–5 pont

2. feladat: Betűkirakó (25 pont)

A. Végeredmény: `AAABAA`

1. kör: `AABBAB`, `ABBBAB`, `ABABAB`, `ABAAAB`, `ABAABB`, `ABAABB`

2. kör: `ABAABB`, `AAAABB`, `AAAABB`, `AAABBB`, `AAABAB`, `AAABAA`

lépésenként 1 pont, ha a végeredmény is jó, akkor +1 13 pont

ha csak a végeredményt adja meg, akkor 5 pont

B. `(10,9)` 4 pont

`(90,9)` 4 pont

`(91,7)` 4 pont

3. feladat: Fa (25 pont)

A. A hívások száma: $2^{db}-1$ 9 pont

Részpontszám: $1+2+4+...+2^{db-1}$ 6 pont

B. A levelek száma: 2^{db-1} 7 pont

C. A lefedett terület: `:db*:h*:v` 9 pont

4. feladat: Mit csinál? (25 pont)

A. `L` = van-e az `A` vektorban legalább `D` darab páros szám 7 pont

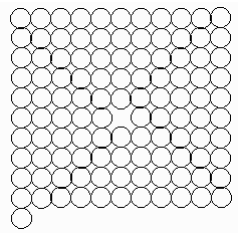
B. Az `A` vektorban az `I`. elem előtt talált páros számok száma 7 pont

C. Legyen az `A` vektorban legalább `D` darab páros szám 6 pont

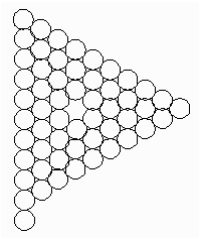
`S` = a `D`. páros szám sorszáma az `A` vektorban 5 pont

Kilencedik-tizedik osztályosok

1. feladat: Karikák (15 pont)

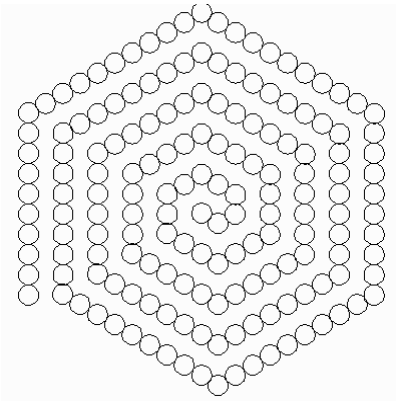
A.  5 pont

Ha lehagyja a bal oldalon lelógó kört, akkor 1 pont levonás.
Ha berajzolja a középről hiányzó kört, akkor 1 pont levonás.

B. 

5 pont

Ha lehagyja a bal oldalon lelógó kört, akkor 1 pont levonás.
Ha berajzolja a középről hiányzó kört, akkor 1 pont levonás.

C. 

5 pont

Ha a spirálisan befelé haladó körök összeérnek, akkor 1 pont levonás.

2. feladat: Halmazvarázs (12 pont)

A. A két halmaz metszetét adja meg `H`-ban. 3 pont

B. Gyorsabb lenne. 1 pont
Ha az egyik halmaz elemei elfogytak, a másikban biztosan nem maradt metszetbeli elem 2 pont

C. Lassúbb lenne. 1 pont
A metszetbeli elemet nem venné ki azonnal a másik halmazból, így azt még egyszer meg kellene vizsgálnia 2 pont

D. Annak a felismeréséért jár pont, hogy elég az egyik halmaz elemeit végignézni a ciklus belsejében (A `Ha nem üres(B) ...` felesleges) 3 pont

3. feladat: Számolgatós (24 pont)

A. `Első`: `N` tizedes jegyre kerekít 5 pont

`Második`: A számhoz 10^{-N}-t ad 5 pont

`Harmadik`: Az `N`. helyi értékig levő számot kivonja 1-ből 5 pont

B. `Első`: hibás, ha `X`=0.99999999995 (`N` db 9-es), vagy ennél nagyobb 3 pont

`Második`: hibás, ha `X`=0.9999999999 (`N` db 9-es) 3 pont

`Harmadik`: hibás, ha `X(N)`=0 3 pont

4. feladat: Cserebere (18 pont)

A. `(B=,=C)` 2 pont

`(A=,=C)` 2 pont

Más megoldás: `(A,C), (B,C), (C=,=C)` 2 pont

B. `(BA,AB)` 1 pont

`(B=,=B)` 1 pont

`(CA,AC)` 1 pont

`(CB,BC)` 1 pont

`(C=,=C)` 1 pont

`(C+,+C)` 1 pont

C. A-val kezdődő sorozatokra:

`(AAB,ABA)` 2 pont

`(BBA,BAB)` 2 pont

`BABABABA` nem alakítható át 2 pont

Egyéb B-vel kezdődő sorozatra:

`(AAB,ABA), (BAA,ABA)` 2 pont

5. feladat: Keresés mátrixban (16 pont)

`AKeres`: A legfelül levő, balról legelső 0 helyét (balról jobbra, felülről lefelé keres) 2+2 pont

`BKeres`: A legjobboldalibb, felülről legelső 0 helyét (felülről lefelé, jobbról balra keres) 2+2 pont

`CKeres`: A legalul levő, jobbról legelső 0 helyét (jobbról balra, alulról felfelé keres) 2+2 pont

Hibásan működnek, ha nincs 0 a mátrixban 4 pont

6. feladat: Kincskereső (15 pont)

A. Az `S` az egyes elágazásokban található pénz értékét tárolja 3 pont

Az `A` logikai tömb a megtett útvonalat írja le: `igaz` volta a bal, `hamis` volta a jobb oldali ág kiválasztását jelenti 3 pont

B. Az N a pénzlelőhelyek száma egy megadott útvonalon 3 pont

C. `f1`: az `A` logikai tömb által leírt útvonal mentén összegyűjthető pénzösszeg 3 pont

`f2`: a legnagyobb pénzlelőhelyhez (ha több ilyen is van, akkor a legközelebbihez) vezető út elágazásainak száma 3 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Síkbeli kód (14 pont)

A. 1: 001, 2: 210, 3: 231, 4: 103 1+1+1+1 pont

B. 12,14,17,18,20 10 pont

Minden jó válasz: +2, minden rossz: –2, az összpont nem lehet negatív.

2. feladat: Halmazok uniója (15 pont)

A. $B \cap C \not\subset A$ (van B-nek és C-nek olyan közös eleme, amely nem eleme A-nak) 3 pont

vagy $A(AN) \not\subset B$ (az A utolsó eleme előfordul B-ben) 2 pont

vagy $A(AN) \notin C$ (az A utolsó eleme előfordul C-ben) 2 pont

B. Mindkét helyen a $<$ helyett $\leq$, a $\geq$ helyett $>$ kell 1+1+1+1 pont

a második ciklusban az A helyett a D halmazt kell vizsgálni, vagy $C(i) \in B$ is megvizsgálandó C összes elemére 4 pont

3. feladat: Bizonytalan (10 pont)

A. Az X, Y, Z változók legnagyobb közös osztója lesz A, B és C értéke 5 pont

B. VALASZ igaz lesz, ha az A mátrixban az X és az Y ugyanabban a sorban van 5 pont

4. feladat: Cserebere (18 pont)

A. `(B=,=C)` 2 pont

`(A=,=C)` 2 pont

Más megoldás: `(A,C), (B,C), (C=,=C)` 2 pont

B. `(BA,AB)` 1 pont

`(B=,=B)` 1 pont

`(CA,AC)` 1 pont

`(CB,BC)` 1 pont

`(C=,=C)` 1 pont

`(C+,+C)` 1 pont

C. A-val kezdődő sorozatokra:

`(AAB,ABA)` 2 pont

`(BBA,BAB)` 2 pont

`BABABABA` nem alakítható át 2 pont

Egyéb B-vel kezdődő sorozatra:

`(AAB,ABA), (BAA,ABA)` 2 pont

5. feladat: Speciális mátrixok (16 pont)

A. `Index(I,J):=N-(I-J)` (vagy `N-I+J`) 4 pont

B. `Index(I,J):=I+J-1` 4 pont

C. `Index(I,J):=I*2+J-2` (vagy `I*3+J-I-2` vagy `(I-1)*3+J-I+1`) 4 pont

D. `Index(I,J):=J*(J-1)/2+I` 4 pont

6. feladat: Ablakok (27 pont)

A. `1,1,-1,-1`. 2 pont

k=1 esetén az ablaktól balra, illetve felfelé 4 pont

k=-1 esetén az ablaktól jobbra, illetve lefelé levő területeket vizsgálja 4 pont

B. Az `x` ablak területe mind a 4 lépésben csökken vagy nem változik `a[m]` és `x` elhelyezkedésétől függően.

1. Az `x` terület: 2,3,5,6,8,9 2 pont
2. Az `x` terület: 5,6,8,9 2 pont
3. Az `x` terület: 5,8 2 pont
4. Az `x` terület: 5 2 pont

C. Az `x` „üres” területté válik, de a vizsgálat folytatódik 2 pont

D. A paraméterként átadott `x` téglalap és az `a[m]` ablak metszete 3 pont

E. `m>0 és x[2]≤x[0] és x[3]≤x[1]` 4 pont

1998. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Rendező-pályaudvar (19 pont)

Ha az első vágányon álló szerelvényhez jön új kocsi, akkor ha az már ész, akkor elindítjuk és új szerelvényt kezdünk, különben pedig hozzácsatoljuk. Ugyanezt tesszük a másik vágánnyal is. Ha az első vágányon levőhöz nem tehetjük és a másodikon nincs még senki, akkor a másodikon kezdünk új szerelvényt. Ha egyik vágányon levőhöz sem tehetjük, akkor a hosszabb szerelvényt elindítjuk, s a helyén kezdünk újat.

```
dbe:=1; dbm:=0; egyik:=érkező(1); másik:=''
Ciklus i=2-től N-ig
  Ha érkező(i)=egyik
     akkor Ha dbe=m akkor Ki: egyik,dbe; dbe:=1
                   különben dbe:=dbe+1
  különben ha érkező(i)=másik
              akkor Ha dbm=m akkor Ki: másik,dbm; dbm:=1
                                 különben dbm:=dbm+1
  különben ha dbm=0 akkor dbm:=1; másik:=érkező(i)
  különben ha dbe<dbm Ki: másik,dbm; dbm:=1; másik:=érkező(i)
  különben Ki: egyik,dbe; dbe:=1; egyik:=érkező(i)
Ciklus vége
Ha dbe≥dbm akkor Ki: egyik,dbe
                 Ha dbm>0 akkor Ki: másik,dbm
különben Ki: másik,dbm; Ki: egyik,dbe
```

Értékelési szempontok

A. Csak egyféle cél, kevesebb, mint M vagon	3 pont
B. Csak egyféle cél, több, mint M vagon	2+2 pont
C. Kétféle cél, kevesebb, mint M vagon	2+2 pont
D. Háromféle cél, az elsőt kell előbb elküldeni, a többit a végén	2+1+1 pont
E. Háromféle cél, a másodikat kell előbb elküldeni, a többit a végén	2+1+1 pont

2. feladat: Állatkertek (27 pont)

Olyan állatot cserélhetnek, amelyből mindkettőnek legalább K darab van:

```
Csere(N,kuk,M,rat,K):
  Ciklus i=1-től N-ig
    Ha kuk(i).db≥K
       akkor j:=1
             Ciklus amíg j≤M és kuk(i).név≠rat(j).név
               j:=j+1
             Ciklus vége
             Ha j≤M és rat(j).db≥K akkor Ki: kuk(i).név
  Ciklus vége
Eljárás vége.
```

Abból lehet ajándékozni, amelyből legalább L példány van, a másik állatkertben pedig egy sem. Ezt az eljárást a feladat megoldásában kétszer kell meghívni, felcserélve a két állatkert szerepét:

```
Ajándék(N,egy,M,mas,L):
  Ciklus i=1-től N-ig
    Ha egy(i).db≥L
       akkor j:=1
             Ciklus amíg j≤M és egy(i).név≠mas(j).név
               j:=j+1
             Ciklus vége
             Ha j>M akkor Ki: egy(i).név
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A. Csak olyat cserél, amiből mindkettőnek van	3 pont
B. Csak olyat cserél, amiből mindkettőnek elég sok van	3 pont
C. Minden olyat cserél, amit lehet	3 pont
D. Kukutyin olyat ajándékoz, amiből neki elég sok van	3 pont
E. Kukutyin olyat ajándékoz, amiből Rátótnak nincs	3 pont
F. Kukutyin mindenből ajándékoz, amiből tud	3 pont
G. Rátót olyat ajándékoz, amiből neki elég sok van	3 pont
H. Rátót olyat ajándékoz, amiből Kukutyinnak nincs	3 pont
I. Rátót mindenből ajándékoz, amiből tud	3 pont

3. feladat: Titkosírás (29 pont)

Első lépésként hozzuk létre a kiterjesztett magyar ábécét és a rákövetkező betűket tartalmazó konstans tömböt:

```
tábla=(('dzs','e'),('cs','d'),('dz','dzs'),('gy','h'),
('ly','m'),('ny','o'),('sz','t'),('ty','u'),('zs','a'),
('a','á'),('á','b'),('b','c'),('c','cs'),('d','dz'),('e','é'),('é
','f'),('f','g'),('g','gy'),('h','i'),('i','í'),('í','j'),
('j','k'),('k','l'),('l','ly'),('m','n'),('n','ny'),('o','ó'),
('ó','ö'),('ö','ő'),('ő','p'),('p','q'),('q','r'),('r','s'),
('s','sz'),('t','ty'),('u','ú'),('ú','ü'),('ü','ű'),('ű','v'),
('v','w'),('w','x'),('x','y'),('y','z'),('z','zs'),('',''));
```

A tömb elemeinek sorrendje nem véletlen, így érjük el, hogy a kétjegyű mássalhangzókat előbb találjuk meg, mint az egyjegyűeket.

Ezután egy szöveg kódolása az alábbi lehet:

```
i:=1; ki:=''
Ciklus amíg i≤hossz(be)
  Keres(i,van,mi)
  Ha van akkor ki:=ki+tábla(mi,2); i:=i+hossz(tábla(mi,1))
  különben ki:=ki+be(i); i:=i+1
Ciklus vége
```

```
Keres(i,van,mi):
  mi:=1; j:=i+hossz(tábla(mi,1))-1
  Ciklus amíg mi<betűszám és be(i..j)≠tábla(mi,1)
    mi:=mi+1; j:=i+hossz(tábla(mi,1))-1
  Ciklus vége
  van:=(mi<betűszám)
Eljárás vége.
```

Értékelési szempontok

A. Az ***angol*** ábécé betűit felismeri és átalakítja 3 betűnként 1 pont (max. 4 pont)

B. Jó betűvé alakítja az ékezet nélküli és az ékezetes magánhangzókat 3+6 pont

C. Kétjegyű betűvé alakít egyjegyű betűket (c→cs, d→dz, g→gy, l→ly, n→ny, s→sz, t→ty, z→zs) 2 betűnként 1 pont (max. 4 pont)

D. Felismeri és átalakítja a többjegyű betűket (cs→d, dz→dzs, dzs→e, gy→h, ly→m, ny→o, sz→t, ty→u, zs→a) 9 pont

E. A nagybetűket, a szóközt és az írásjeleket nem változtatja meg 3 pont

Kilencedik-tizedik osztályosok

1. feladat: Vetésterület (16 pont)

A bal felső sarokban mindenképpen indul egy tábla, s belátható, hogy választhatunk bármely olyan téglalapot táblának, amely sem sorral, sem oszloppal nem bővíthető, azzal a táblák számára ugyanazt a megoldást kapjuk.

```
db:=0
Ciklus i=1-től N-ig
  Ciklus j=1-től M-ig
    Ha b(i,j)≠'' akkor db:=db+1; Számolás(i,j,di,dj)
                       Ciklus k=i-től di-ig
                         Ciklus l=j-től dj-ig
                           b(k,l):=''
                         Ciklus vége
                       Ciklus vége
  Ciklus vége
Ciklus vége
```

Először az (i,j) pozícióból megpróbálunk lefelé menni, amíg csak lehet (azaz azonos növényeket találunk), majd az így kapott területet jobbra újabb oszlopokkal bővíteni, szintén amíg csak lehet:

```
Számolás(i,j,di,dj):
  di:=i; dj:=j
  Ciklus amíg di<n és b(di+1,j)=b(i,j)
    di:=di+1
  Ciklus vége
  Ciklus amíg dj<m és Jóoszlop(i,j,di,dj+1)
    dj:=dj+1
  Ciklus vége
Eljárás vége.
```

```
Jóoszlop(i,j,di,dj):
  s:=i
  Ciklus amíg s≤di és b(i,j)=b(s,dj)
    s:=s+1
  Ciklus vége
  Jóoszlop:=s>di
Függvény vége.
```

Értékelési szempontok

A. Homogén területet felismer	2 pont
B. Két téglalapot felismer	4 pont
C. L-alakú területet jól kezel	6 pont
D. Egy mező közepén egyetlen más növényből álló folt	4 pont

2. feladat: Határőrség (25 pont)

Előreolvasunk minden határállomásról egy rekordot, majd amíg az összes file nem ürül ki, addig kiválasztjuk a legkisebb dátumút, megkeressük annak az előző határátlépését. Ha még nem volt sehol, akkor tároljuk, ha volt már valahol, akkor pedig ellenőrizzük, hogy követett-e el határsértést. A feldolgozás végén pedig megnézzük, hogy ki ment úgy külföldre, hogy az év végéig még nem jött vissza.

```
Ciklus i=1-től N-ig
  Nyit(f(i),határ(i))
  Ha vége?(f(i)) akkor átlépő(i).idő:=1000
              különben Olvas(f(i),átlépő(i))
Ciklus vége
db:=0
Ciklus amíg vanmégadat
  Legkisebbdátumú(i,n); Előzőadatkeresés(i,db,van,j)
  Ha van akkor Ha átlépő(i).irány=múlt(j).irány
                  akkor Határsértő(átlépő(i).név)
               különben múlt(j):=átlépő(i)
       különben db:=db+1; múlt(db):=átlépő(i)
  Ha vége?(f(i)) akkor átlépő(i).idő:=1000
              különben Olvas(f(i),átlépő(i))
Ciklus vége
Ciklus i=1-től db-ig
  Ha múlt(i).irány='KI' akkor Külföldön(múlt(i).név)
Ciklus vége
```

A `Határsértő`, illetve a `Külföldön` eljárás a kért formátumban írja ki a paraméterét.

```
Legkisebbdátumú(i,n)
  i:=1
  Ciklus j=2-től N-ig
    Ha átlépő(i).idő>átlépő(j).idő akkor i:=j
  Ciklus vége
Eljárás vége.
```

```
Előzőadatkeresés(i,db,van,j):
  j:=1
  Ciklus amíg j≤db és átlépő(i).név≠múlt(j).név
    j:=j+1
  Ciklus vége
  van:=(j≤db)
Eljárás vége.
```

Értékelési szempontok

A. Üres állományokat tud kezelni	1 pont
B. Egy határátkelőhely esetén jól adja meg a külföldön levőket	3 pont
C. Egy határátkelőhely esetén jól adja meg az egymás után kétszer be- vagy kilépőket	2-2 pont
D. Nem zavarja, ha valaki belépéssel kezdi	2 pont
E. Tud több határátkelőhelyet kezelni	2 pont

F. Jól kezeli az egyiken be- és a másikon ki-, majd újra az egyiken belépőket (nem mondja határsértőnek) 3 pont

G.Felismeri a különböző határátkelőhelyeken határsértőket 3-3 pont

H. Tudja kezelni 2 lakos 3000 határátlépését 4 pont

3. feladat: Kemping (17 pont)

Első lépésként határozzuk meg, hogy mely napoktól kezdődően hány igénylés van:

```
igény:=(0,...,0)
Ciklus i=1-től N-ig
  igény(x(i)):=igény(x(i))+1
Ciklus vége
```

Ezután mohó stratégiával végignézzük az igényléseket, s mindegyiket a legelső faházba tesszük, amelyik szabad. Átlépjük azokat, amelyek nem teljesíthetőek.

```
Megold:=0; kövszabad:=(0,...,0)
Ciklus i=1-től 365-ig
  vanhely:=igaz;
  Ciklus amíg igény(i)>0 és vanhely
    Szabadkeresés(i,j,vanhely)
    Ha vanhely akkor Megold:=Megold+1; igény(i):=igény(i)-1
                     kövszabad(j):=i+M
  Ciklus vége
Ciklus vége.
```

```
Szabadkeresés(i,j,vanhely):
  j:=1
  Ciklus amíg j≤K és kövszabad(j)>i
    j:=j+1
  Ciklus vége
  vanhely:=(j≤K)
Eljárás vége.
```

Értékelési szempontok

A. Az igények nem ütköznek, és egy faház van 1 pont

B. Az igények nem ütköznek, és két faház van 2 pont

C. Az igények ütköznek, és egy faház van 3 pont

D. Az igények ütköznek, és több faház van 3 pont

E. K=100 és N=1000, véletlen 4 pont

F. K=10 és N=1000, véletlen 4 pont

4. feladat: Mars-kutatás (17 pont)

A marsjárók olyan pontokról gyűjthetik be a kőzetmintát, amelyekre az (1,1) pontból eljuthatnak, és amelyekről az (M,N) pontba eljuthatnak.

Első lépésként határozzuk meg, hogy mely pontokba vezet út az (1,1) pontból: általában igaz az, hogy az (i,j) pontba csak az (i,j-1) vagy az (i-1,j) pontból jöhetünk (a mátrix két szélét külön kell kezelnünk):

```
E11(1,1):=igaz
Ciklus i=2-től M-ig
  E11(i,1):=E11(i-1,1) és T(i,1)≠2
Ciklus vége
Ciklus j:=2-től N-ig
  E11(1,j):=E11(1,j-1) és T(1,j)≠2
Ciklus vége
Ciklus i=2-től M-ig
  Ciklus j=2-től N-ig
    E11(i,j):=T(i,j)≠2 és (E11(i,j-1) vagy E11(i-1,j))
  Ciklus vége
Ciklus vége
```

Ugyanígy határozható meg, hogy mely pontokból lehet eljutni az (M,N) pontba, csak visszafelé kell haladnunk:

```
EMN(M,N):=igaz
Ciklus i=M-1-től 1-ig -1esével
  EMN(i,N):=EMN(i+1,N) és T(i,N)≠2
Ciklus vége
Ciklus j=N-1-től 1-ig -1-esével
  EMN(M,j):=EMN(M,j+1) és T(M,j)≠2
Ciklus vége
Ciklus i=M-1-től 1-ig -1-esével
  Ciklus j=N-1-től 1-ig -1-esével
    EMN(i,j):=T(i,j)≠2 és (EMN(i,j+1) vagy EMN(i+1,j))
  Ciklus vége
Ciklus vége
```

Ezután a feladat megoldása: számoljunk minden olyan 1-es értékű elemnél, amelyikhez a két előállított mátrixban igaz érték tartozik:

```
Meg:=0
Ciklus i=1-től M-ig
  Ciklus j=1-től N-ig
    Ha E11(i,j) és EMN(i,j) és T(i,j)=1 akkor Meg:=Meg+1
  Ciklus vége
Ciklus vége
```

Értékelési szempontok

A. Nincs sziklás terület	1 pont
B. Sziklás terület kőzetet fog körül	3 pont
C. Sziklás terület csak belső pontokban van	3 pont
D. Általános eset, M,N=10	5 pont
E. Általános eset, M,N=100	5 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Szálloda (14 pont)

Első lépésként a beolvasott szöveges kérdéseket fel kell bontani elemekre:

```
Szétszed(szöveg,mit,rel,osztály,minmax):
  Ha szöveg(1..8)='SZAKACS:' akkor mit:=1; tól:=9
  különben ha szöveg(1..8)='CUKRASZ:' akkor mit:=2; tól:=9
  különben mit:=3; tól:=11
  Elágazás szöveg(tól) szerint
    '<': Ha szöveg(tól+1)='=' akkor rel:=4 különben rel:=1
    '>': Ha szöveg[tól+1)='=' akkor rel:=5 különben rel:=2;
    '=': rel:=3
  Elágazás vége
  Ha rel>3 akkor tól:=tól+2 különben tól:=tól+1
  i:=keres(':',szöveg,tól); számmá(szöveg(tól..i),osztály)
  minmax:=(szöveg(i..i+2)='MIN')
Eljárás vége.
```

A megoldás lényege ezek után egy a `mit` paramétertől függő keresés meghívása:

```
Olvas(f,szöveg)
Szétszed(szöveg,mit,rel,osztály,minmax)
Ha mit≠2 akkor Keres(n,szakács,rel,osztály,minmax,van1,s1)
Ha mit≠1 akkor Keres(m,cukrász,rel,osztály,minmax,van2,s2)
Kiírás
```

A kiírásban persze figyelemmel kell lenni arra, hogy mit kértünk, illetve, hogy van-e a feltételnek megfelelő szakács, illetve cukrász. Figyelni kell arra is, hogy ha bármelyikük jó, akkor melyiket találtunk, s ha mindkettőt, akkor még választani kell közöttük.

A keresés a szövegből kiszedett feltételek szerint keres meg egy elemet a `szakács` vagy a `cukrász` tömbben. Ehhez meg kell keresni az első adott tulajdonságú elemet, majd a továbbiak közül kel kiválasztani a relációnak megfelelő adott tulajdonságút!

```
Keres(n,miben,rel,osztály,minmax,van,s)
  s:=1
  Ciklus amíg s≤n és nem tul(s)
    s:=s+1
  Ciklus vége
  Ciklus i=s+1-től N-ig
    Ha tul(i) és rel(i,s) akkor s:=i
  Ciklus vége
  van:=(s≤n)
Eljárás vége.
```

```
tul(i):
  Elágazás rel szerint
    1: tul:=(miben(i).oszt<osztály)
    2: tul:=(miben(i).oszt>osztály)
    3: tul:=(miben(i).oszt=osztály)
    4: tul:=(miben(i).oszt<=osztály)
    5: tul:=(miben(i).oszt>=osztály)
  Elágazás vége
Függvény vége.
```

```
rel(i,s):
  Ha minmax akkor rel:=(miben(i).év<miben(s).év vagy
                        miben(i).év=miben(s).év és
                        miben(i).név≤miben(s).név)
          különben rel:=(miben(i).év>miben(s).év vagy
                        miben(i).év=miben(s).év) és
                        miben(i).név≤miben(s).név)
Függvény vége.
```

Értékelési szempontok

A. Szakácsot tud keresni	1 pont
B. Mindkettőt tud keresni	1 pont
C. Cukrászt tud keresni	1 pont
D. Minden relációt ismer (<,>,=,≤,≥)	1-1-1-1-1 pont
E. MAX-ot és MIN-t tud keresni	2-2 pont
F. A névsorban legelsőt adja az egyformák közül	2 pont
G. Felismeri, ha nincs a keresett	1 pont

2. feladat: Radar (20 pont)

A főprogramban a beolvasott észlelések alapján kell elvégezni a feldolgozást. Ha egy rakétát már észleltünk, akkor a későbbi riasztás elkerülése érdekében minden további észlelésének idejét lenullázzuk:

```
Ciklus i=1-től n-ig
  Ha észlel(i).idő>0 akkor Feldolgoz(i)
Ciklus vége
```

```
Feldolgoz(i):
  j:=i+1; kész:=hamis
  Ciklus amíg nem kész és j≤n
    tj:=észlel(j).idő-észlel(i).idő
    Ha tj≥9 és jó(i,j,tj)
       akkor k:=j+1
             ts:=(észlel(j).sor-észlel(i).sor) div tj
             to:=(észlel(j).oszl-észlel(i).oszl) div tj
             Ciklus amíg nem kész és k≤n
               tk:=észlel(k).idő-észlel(j).idő
               Ha tk≥9 és jó(j,k,tk) és jó2(j,k,tk,ts,to)
                  akkor kész:=igaz; riaszt(i,j,k,ts,to)
               k:=k+1
             Ciklus vége
    j:=j+1
  Ciklus vége
Eljárás vége.
```

Ha időegységenként a rakéta koordinátája egész számmal változik, akkor `tj` időegység alatt `tj`-vel osztható egész számmal kell változnia. Ezt vizsgálja a következő függvény:

```
jó(i,j,tj):
  jó:=(|észlel(j).sor-észlel(i).sor| mod tj=0 és
       |észlel(j).oszl-észlel(i).oszl| mod tj=0)
Függvény vége.
```

A harmadik megfigyelésre ugyannyival kell változnia a koordinátáknak, mint a második megfigyelése változott:

```
Jó2(i,j,tj,mi1,mi2):
  jó:=(|észlel(j).sor-észlel(i).sor| div tj=mi1 és
       |észlel(j).oszl-észlel(i).oszl| div tj=mi2)
Függvény vége.
```

Csak azon rakéták esetén kell riasztani, amelyek a bázison csapódnának be, s ezek összes észlelését le kell tiltani a következő vizsgálatokból!

```
Riaszt(i,j,k,ts,to):
  Ha célbaér(i,j,k,ts,to,bidő,bs,bo)
     akkor Ki: észlel(k).idő,bidő,bs,bo
           Ciklus ii:=i+1-től N-ig
             t:=észlel(ii).idő-észlel(i).idő
             Ha t≥9 és jó2(i,ii,t,ts,to)
                akkor észlel(ii).idő:=0
           Ciklus vége
Eljárás vége.
```

A célbaérés úgy vizsgálható, hogy időegységenként követjük a rakétát mindaddig, amíg vagy ki nem ér a megfigyelt területről, vagy pedig a célterület fölé ér.

```
célbaér(i,j,k,ts,to,bidő,bs,bo):
  bs:=észlel(i).sor; bo:=észlel(i).oszl
  bidő:=észlel(i).idő
  Ciklus amíg nem célban(bs,bo) és nem kívül(bs,bo)
    bidő:=bidő+1; bs:=bs+ts; bo:=bo+to
  Ciklus vége
  célbaér:=célban(bs,bo)
Függvény vége.
```

```
célban(sor,oszlop):
  célban:=(sor≥496 és sor≤505 és oszlop≥496 és oszlop≤505
Függvény vége.
```

```
kívül(sor,oszlop):
  kívül:=(sor<1 vagy sor>1000 vagy oszlop<1 vagy oszlop>1000)
Függvény vége.
```

Értékelési szempontok

A. Négy irányból érkező rakéták miatt riaszt	2-2-2-2 pont
B. Két elkerülő és egy átrepülő, de nem becsapódó rakéta miatt nem riaszt	2-2 pont
C. Sorozatindítás ugyanarról a helyről, mindet jelzi	1-1-1-1 pont
D. Négy irányból érkező, de lelőtt rakéták miatt riaszt	1-1-1-1 pont

3. feladat: Járműkölcsönzés (11 pont)

Jelöljük F(i)-vel, hogy az év i-edik napjáig mi a lehető legjobb megoldás! A feladat dinamikus programozás módszerével oldható meg: az i-edik napig az optimális megoldás vagy az i-1-edik napig tartó optimális megoldás, vagy pedig azon igények kezdete előtti megoldás megnövelve az igény hosszával, amelyek pontosan az i-edik napon fejeződnek be:

```
F(0):=0
Ciklus i=1-től Ev-ig
  Max:=F(i-1)
  Ciklus j=1-től N
    Ha igény(j).B=i
       akkor Uj:=F(igény(j).A-1)+igény(j).B-igény(j).A+1
             Ha Uj>Max akkor Max:=Uj
    Elágazás vége
  Ciklus vége
  F(i):=Max
Ciklus vége
Megold:=F(Ev)
```

Értékelési szempontok

A. Nincs ütköző megrendelés	1 pont
B. Van ütközés, de a jármű teljesen kihasználható	2 pont
C. A maximális megrendelések kielégítése az optimum	3 pont
D. Általános eset, N=20	2 pont
E. Általános eset, N=1000	3 pont

4. feladat: Robot (12 pont)

Keressük meg az aktuális oszlopban a legalsó 1-est! Idáig egy robotnak biztos le kell jönnie, a korábbi oszlopokból a leglejjebb, de ennél a pontnál feljebb levő átjöhet ezt az 1-est is felszedni, s haladhatunk az oszlopban felfelé. Ha újabb 1-est találunk, akkor meg kell nézni, hogy az előző oszlopokból jöhet-e át ide valamelyik robot, … és így tovább.

```
Ciklus i=2-től M-ig
  U(i):=0
Ciklus vége
U(1):=1
Ciklus j=1-től N-ig
  i:=M+1; U(0):=1
  Ciklus amíg i>0
    Ciklus
      i:=i-1
    amíg i>0 és T(i,j)≠1
    Ciklus vége
    Ha i>0 akkor ii:=i
                 Ciklus amíg U(ii)=0
                   ii:=ii-1
                 Ciklus vége
                 U(ii):=0; U(i):=1; i:=ii
  Ciklus vége
Ciklus vége
Megold:=0
Ciklus i=1-től M-ig
  Megold:=Megold+U(i)
Ciklus vége
```

Az alábbi ábrán vastagon szedve láthatjuk a mintapéldában a megoldás során követett utakat:

```
0-1-1-1-0-0-1-0-1 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0
0 0 0 1-0 0 0 0 0 0 0 0
0 1 0 0 1-0 0 0 0 0 0 0
1 0 0 0 0 1-1-1-1 0 0 0
1 1-1-1-1-1-0-0-0-1 0 0
1 0 0 0 0 0 0 0 0 1 0 0
1-0-0-0-0-0-0-0-0-1-0-0
```

Értékelési szempontok

A. Egyetlen sorban van csak gép	1 pont
B. Egyetlen oszlopban van csak gép	1 pont
C. Minden sorban egy gép van, átlós elrendezésben	2 pont
D. Általános eset, M,N=10	4 pont
E. Általános eset, M,N=100	4 pont

5. feladat: Fordítóprogram (18 pont)

A fordítást két menetben végezzük. Az első menetben konstansokkal töltjük a memóriát és kiszámoljuk a címkék memóriacímét.

A címke után, ha van még a sorban valami, akkor öt esetet kell megvizsgálnunk!

Ha aposztróf az első karakter, akkor a második karaktert kell a zárójelbe tett számszor a memóriába írnunk! Ha idézőjel az első, akkor pedig az idézőjelek közötti szöveget kell a memóriába írni! Mindkét esetben a beírt karakterek mögé kell lépni!

A [-jel, illetve < jel után a memóriában 1-gyel, illetve 2-vel kell továbblépni! Ha pedig számjeggyel indul a sor, akkor a számot kell a zárójelben levő számszor a memóriába írni!

```
Menet1:
  Nyit(f); db:=0; cím:=1
  Ciklus amíg nem vége?(f)
    Sorolvasás(f,sor); Szóközkihagyás(sor)
    Keresés(':',sor,i,van)
    Ha van akkor db:=db+1; címke(db).neve:=sor(1..i-1)
                 címke(db).címe:=cím
                 sor:=sor(i+1..hossz(sor))
    Elágazás vége
```

```
    Ha sor≠'' akkor
       Elágazás sor(1) szerint
         #39: Keresés('(',sor,i,van)
              Számmá(sor(i+1..hossz(sor)-1),j)
              Ciklus m:=1-től j-ig
                p(cím+m-1):=érték(sor(2))
              Ciklus vége
              cím:=cím+j
         '"': Ciklus m:=1-től hossz(sor)-2-ig
                p(cím+m-1):=érték(sor(m+1))
              Ciklus vége
              cím:=cím+hossz(sor)-2
         '[': cím:=cím+1
         '<': cím:=cím+2
       különben Keresés('(',sor,i,van)
                Számmá(sor(i+1..hossz(sor)-1),j)
                Számmá(sor(1..i-1),e); b:=e és 255
                Ciklus m:=1-től j-ig
                  p(cím+m-1):=b
                Ciklus vége
                cím:=cím+j
       Elágazás vége
  Ciklus vége
  Zár(f)
Eljárás vége.
```

A második menetben a legtöbb esetben csak a memóriacímet növeljük a szükséges mértékben. Két esetben pedig meg kell keresni, hogy honnan kell értéket másolni!

```
Menet2:
  Nyit(f); cím:=1
  Ciklus amíg nem vége(f)
    Sorolvasás(f,sor); Szóközkihagyás(sor)
    Keresés(':',sor,i,van)
    Ha van akkor sor:=sor(i+1..hossz(sor))
    Ha sor≠'' akkor
       Elágazás sor(1) szerint
         #39: Keresés('(',sor,i,van)
              Számmá(sor(i+1..hossz(sor)-1),j); cím:=cím+j
         '"': cím:=cím+hossz(sor)-2
         '<': keres(sor,címe); p(cím):=címe mod 256
              p(cím+1):=címe div 256; cím:=cím+2
         '[': keres(sor,címe); p(cím):=p(címe); cím:=cím+1
       különben Keresés('(',sor,i,van)
                Számmá(sor(i+1..hossz(sor)-1),j)
                cím:=cím+j
       Elágazás vége
  Ciklus vége
  Zár(f)
Eljárás vége.
```

```
keres(sor,címe):
  sor:=sor(2..hossz(sor)-1); keresés(' ',sor,i,van)
  Ha nem van akkor i:=hossz(sor)
  sor:=sor(1,i-1); címe:=1
  Ciklus amíg sor≠címke(címe).neve
    címe:=címe+1
  Ciklus vége
  címe:=címke(címe).címe
Eljárás vége.
```

Értékelési szempontok

A. `x(DB)` utasítást jól fordítja (érték+darabszám)	1+1 pont
B. `'x'(DB)` utasítást jól fordítja (érték+darabszám)	1+1 pont
C. `"xxxxxx"` utasítást jól fordítja	2 pont
D. `<X>` utasítást jól fordítja, és jó sorrendbe teszi a 2 byte-ot	1+1 pont
E. `[Y]` utasítást jól fordítja	2 pont
F. `Címke:` címkedefiníciót jól fordítja	2 pont
G. Tud előrehivatkozást is fordítani	2 pont
H. Szóközöket szűri	2 pont
I. ‘‘ és " " belsejében a szóközöket nem szűri	1+1 pont

1998. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Méréskiegészítés (25 pont)

Külön meg kell vizsgálni azt az esetet, amikor a méréssorozat elején, illetve végén hiányoznak mérések, valamint azt, amikor a közepén:

```
i:=1
Ciklus amíg i≤n és mérés(i)=-1000
  i:=i+1
Ciklus vége
Ha i>1 és i≤n akkor Elejére(i)
e:=i; i:=n
Ciklus amíg i≥1 és mérés(i)=-1000
  i:=i-1
Ciklus vége
Ha i≥1 és i<n akkor Végére(i)
u:=i
Ciklus i:=e-től u-ig
  Ha mérés(i)=-1000 akkor Közepére(i)
Ciklus vége
```

```
Elejére(i):
  Ciklus j=1-től i-1-ig
    mérés(j):=mérés(i)
  Ciklus vége
Eljárás vége.
```

```
Végére(i):
  Ciklus j=i+1-től n-ig
    mérés(j):=mérés(i)
  Ciklus vége
Eljárás vége.
```

Míg a két szélére könnyű volt a szélső helyes elem ismételt elhelyezése, a közepén meg kell számolni, hogy hány mérés hiányzik, ki kell számolni a hiányzók két szomszédjából a növekedés (vagy csökkenés) lépésközét, majd ebből számolhatjuk a hiányzó értékeket:

```
Közepére(i):
  j:=i
  Ciklus amíg mérés(j)=-1000
    j:=j+1
  Ciklus vége
  lépés:=(mérés(j)-mérés(i-1))/(j-i+1)
  Ciklus k=i-től j-1-ig
    mérés(k):=mérés(k-1)+lépés
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A. Jól írja ki a változatlan adatokat	4 pont
B. Jól becsüli a bal és a jobb szélen hiányzó egyetlen adatot	2-2 pont
C. Jól becsüli a bal és a jobb szélen hiányzó adatokat	3-3 pont
D. Jól becsüli a hiányzó adatot, ha a szomszédjai ismertek	4 pont
E. Jól becsül hosszabb hiányzó szakaszon	7 pont

2. feladat: Karácsonyfa (25 pont)

Az A részfeladat megoldása egy minimumkiválasztás:

```
Afeladat:
  min:=1
  Ciklus i=2-től K-ig
    Ha f(i).ár<f(min).ár akkor min:=i
  Ciklus vége
  legolcsóbb:=min
Eljárás vége.
```

A B részfeladatban meg kell nézni, hogy milyen fák fordulnak elő (a további részfeladatok előkészítéséhez tároljuk ezek nevét és darabszámát is):

```
Bfeladat:
  fadb:=0
  Ciklus i=1-től K-ig
    j:=1
    Ciklus amíg j≤fadb és fajták(j).név≠f(i).név
      j:=j+1
    Ciklus vége
    Ha j>fadb akkor fadb:=fadb+1; fajták(fadb).db:=1
                    fajták(fadb).név:=f(i).név
           különben fajták(j).db:=fajták(j).db+1
  Ciklus vége
Eljárás vége.
```

A C részfeladat megoldása az előző részben kiszámolt `fajták` tömb `fadb` darab eleme. A D részfeladatban `fadb` darab minimumkiválasztást kell alkalmazni, amihez felhasználjuk a fajták tömböt is.

```
Dfeladat:
  f(0).ár:=+∞
  Ciklus j=1-től fadb-ig
    min:=0
    Ciklus i:=1-től K-ig
      Ha f(i).név=fajták(j).név és f(i).ár<f(min).ár
         akkor min:=i
    Ciklus vége
    d(j):=f(min)
  Ciklus vége
Eljárás cége.
```

Értékelési szempontok

A részfeladat	4 pont
B részfeladat	5 pont
C részfeladat	8 pont
D részfeladat	8 pont

3. feladat: Karaktersorozat keresése (25 pont)

A két szót nagybetűsre váltjuk C feltétel esetén, majd a keresést minden lehetséges pozíción elvégezzük. A B feltétel esetén nézzük, hogy teljes szót találtunk-e.

```
Be: mit,miben,feltétel
Ha feltétel(2)='C' akkor Nagyravált
Ciklus i=1-től hossz(miben)-hossz(mit)+1-ig
  Ha miben(i..i+hossz(mit)-1)=mit
     akkor Ha feltétel(1)='B' akkor Ki: i
           különben ha Szóeleje(i) és Szóvége(i+hossz(mit)-1)
                    akkor Ki: i
Ciklus vége

Szóeleje(i):
  Ha i=1 akkor Szóeleje:=igaz
      különben Szóeleje:=miben(i-1)=' '
Függvény vége.

Szóvége(i):
  Ha i=hossz(miben) akkor Szóvége:=igaz
  különben Szóvége:=miben(i+1)∈(' ','.',',',';',':','!','?')
Függvény vége.
```

Értékelési szempontok

A. Tud karaktersorozatot keresni (abcd, ababcdab, BC → 3)	3 pont
B. Tud teljes szót keresni (abcd, abcde abcd aabcd, AC → 7)	3 pont
C. Megkülönbözteti a kis- és nagybetűket, ha kell (abcd, Abcd abcd aabcd, AC → 6)	4 pont
D. Szóhatár írásjel is lehet (abcd, abc abcd. abcd! ab, AD → 5,11)	4 pont
E. Kis- és nagybetűk megkülönböztetése nélkül keres (abc, ABC ab, AC → 1)	3 pont
F. Kis- és nagybetűk megkülönböztetése nélkül keres (abc, aAbc ab, BC → 2)	3 pont
G. Az összeset megadja (abc, ab abc ab abc ab abc ABC, AD → 4,11,18)	5 pont

Kilencedik-tizedik osztályosok

1. feladat: Ütemező (50 pont)

A megoldás két fő lépésből áll. Az elsőben beolvassuk a file tartalmát és generáljuk az `EGbegin` `EGend` közötti részéből a végrehajtható ESEM nyelvű programot:

```
Ciklus amíg nem vége?(f)
  Olvas(f,sor); Szóközszűrés(sor)
  Leválaszt(sor,k,l); Generál(k,l)
Ciklus vége
```

A második részben ezután végrehajtjuk a programot T időegységig. A megoldás lényege: kiválasztjuk a legkisebb időpontú eseményt, majd azt végrehajtjuk:

```
idő:=0.0; vanmég:=igaz
Ciklus amíg idő≤T és vanmég
  Minkiválaszt(idő,j,vanmég)
  Ha vanmég akkor Ki: egészrész(idő),gen(j).név)
Ciklus vége
```

A generálás minden eseményhez kezdő- és végidőt generál; megadja, hogy mely esemény előtt, illetve után kell végrehajtani; beállítja, hogy ez az esemény még nem fordult elő. Ha `EGbegin` utasításhoz ér, akkor rekurzívan meghívja magát. Ha `EGend`-hez ér, akkor pedig befejeződik.

```
Generál(kezd,lép):
  Ciklus
    Olvas(f,sor); Szóközszűrés(sor)
    Ha sor(1..7)='EGbegin'
       akkor Leválaszt(sor,k,l); Generál(kezd+k*lép,l*lép)
    különben ha sor(1..5)≠'EGend'
       akkor Felismer(sor,n,m,kk,név,előző,következő)
             gendb:=gendb+1; gen(gendb).név:=név
             gen(gendb).kezd:=kezd+n*lép
             gen(gendb).lépés:=kk*lép
             gen(gendb).vég:=kezd+m*lép
             gen(gendb).előző:=előző
             gen(gendb).következő:=következő
             gen(gendb).volt:=hamis
  amíg sor(1..5)≠'EGend'
  Ciklus vége
Eljárás vége.
```

A beolvasott sor biztosan számmal kezdődik. Ha ezt nem + jel követi, akkor az első típusú eseményről van szó. A + jelet biztosan egy másik szám követi. Ha ezután nem < jel van, akkor az esemény második típusú. A < jel után pedig biztosan harmadik számnak kell következni a harmadik típusú eseményhez.

Ezek után következik az esemény neve, amit követhet egy < vagy egy > jellel bevezetett újabb eseménynév.

```
Felismer(sor,n,m,k,név,előző,következő):
  i:=1
  Ciklus amíg sor(i)∈('0'..'9')
    i:=i+1
  Ciklus vége
  Számmá(sor(1..i-1),n)
```

```
  Ha sor(i)≠'+' akkor m:=n+1; k:=1
  különben j:=i+1
           Ciklus amíg sor(j)∈('0'..'9')
             j:=j+1
           Ciklus vége
           Számmá(sor(i+1..j-1),k)
           Ha sor(j)≠'<' akkor m:=+∞
           különben i:=j+1
                    Ciklus amíg sor(i)∈('0'..'9')
                       i:=i+1
                    Ciklus vége
                    Számmá(sor=j+1..,i-1),m)
           Elágazás vége
  Elágazás vége
  név:=''
  Ciklus amíg i≤hossz(sor) és sor(i)∉{'<',’>’}
    név:=név+sor(i); i:=i+1
  Ciklus vége
  előző:=''; következő:=''
  Ha i<hossz(sor)
     akkor nnév:=sor(i..hossz(sor))
           Ha sor(i-1)='<' akkor előző:=nnév
                           különben következő:=nnév
  Elágazás vége
Eljárás vége.
```

Az aktuális idő előtt kezdődött események közül azok, amelyeknek még nem jött el a végideje, egy lépéssel továbbléptethetők. Válasszuk ki ezután a legkisebb olyan kezdőidejű eseményt, amely az `idő` időpillanatban végrehajtható és volt már az az esemény, ami után következhet! Ezután azok az eseményeket meg kell szüntetni, amik csak az így kiválasztott esemény előtt keletkezhetnek!

```
Minkiválaszt(idő,j,vanmég):
  gen(0).kezdet:=+∞; min:=0
  Ciklus i=1-től gendb-ig
    Ha gen(i).kezd<idő
       akkor Ha gen(i).vég>idő
                akkor gen(i).kezd:=gen(i).kezd+gen(i).lépés
             különben gen(i).kezd:=+∞
    Elágazás vége
    Ha gen(i).kezd<gen(i).vég és voltmár(gen(i).következő)
       és gen(min).kezd>gen(i).kezd akkor min:=i
  Ciklus vége
  Ciklus i=1-től gendb-ig
    Ha gen(i).előző=gen(min).név akkor gen(i).kezd:=+∞
  Ciklus vége
  j:=min; idő:=gen(min).kezd; gen(min).volt:=igaz
  gen(min).kezd:=gen(min).kezd+gen(min).lépés
  vanmég:=(min≠0 és idő≤T)
Eljárás vége.
```

Ha egy eseménynek nincs előfeltétele, akkor a `voltmár` függvény legyen igaz értékű, különben pedig nézzük meg, hogy előfordult-e már!

```
voltmár(név):
  Ha név='' akkor voltmár:=igaz
  különben i:=1
           Ciklus amíg név≠gen(i).név
             i:=i+1
           Ciklus vége
           voltmár:=gen(i).volt
Függvény vége.
```

Értékelési szempontok

A. Egyetlen eseménygenerátor		**10 pont**
Jól kezeli az üres eseménygenerátort		2 pont
Egyetlen eseménygenerátort jól kezel		8 pont
n esetben	2 pont	
n+k<n esetben	3 pont	
n+k esetben	3 pont	
B. Kapcsolat eseménygenerátorok között		**22 pont**
Két azonos szinten levő eseménygenerátort jól kezel		2 pont
Jól kezeli a kisebb (/L) lépésközt		5 pont
Jól kezeli a nagyobb (*L) lépésközt		5 pont
Jól kezeli a más esemény után bekövetkezőket		5 pont
Jól kezeli a más esemény előtt bekövetkezőket		5 pont
C. Egymásba ágyazott eseménygenerátorok		**12 pont**
Az egymásba ágyazottakat jól kezeli, ha azonos a lépésköz		2 pont
Az egymásba ágyazottakat jól kezeli, ha nagyobb a lépésköz		2 pont
Az egymásba ágyazottakat jól kezeli, ha kisebb a lépésköz		2 pont
Több mélységű egymásba ágyazást és alsóbb szinten előforduló párhuzamosságot is kezel		3+3 pont
D. Nagyon sok eseményt (>10 000) is jól kezel		6 pont

2. feladat: Karaktersorozat keresése a Weben (25 pont)

Az adott file-ban (a `név` tömb első eleme) kezdve meg kell keresni a beolvasott szövegrészeket a feltételnek megfelelően (a `keresés` eljárás ki is írja a megfelelő adatokat):

```
Ciklus amíg nem vége?(f)
  Olvas(f,mit); Ha feltétel(2)='C' akkor Nagyravált(mit)
  névdb:=1; Keresés(mit)
Ciklus vége
```

Az összes állományt, amiben keresni kell, megnyitjuk, végigolvassuk és a sorai adott hosszú részeit a feltételnek megfelelően hasonlítjuk a keresett szöveggel. Figyelnünk kell arra, hogy a beolvasott sorban újabb állományra történhet hivatkozás!

```
Keresés(mit)
  i:=1
  Ciklus amíg i≤névdb
    kell:=igaz; j:=0; Nyit(h,név(i))
    Ciklus amíg nem vége?(h)
      Olvas(h,sor)
      Ha feltétel(2)='C' akkor Nagyravált(sor)
      Hivatkozás(sor)
      k:=1
      Ciklus amíg k≤hossz(sor)
        j:=j+1
        Ha k≤hossz(sor)-hossz(mit)+1
           akkor Ha sor(k..k+hossz(mit)-1)=mit
                    akkor Ha van(sor,k,mit) akkor Kiir(j)
        k:=k+1
      Ciklus vége
    Ciklus vége
    Zár(h)
    Ha nem kell akkor Sorvégírás
    i:=i+1
  Ciklus vége
Eljárás vége.

van(sor,k,mit):
  Ha feltétel(1)='B' akkor van:=igaz
  különben van:=Szóeleje(sor,k) és Szóvége(sor,k+hossz(mit)-1)
Függvény vége.

Kiír(k):
  Ha kell akkor Kiír(név(i)); kell:=hamis
  Kiír(k)
Eljárás vége.
```

Hivatkozás olyan sorokban lehet, amelyekben van &-jel, s ekkor az új állomány neve a következő & jelig tart (de csak akkor kell felvenni az állományok közé, ha még nem volt rá hivatkozás):

```
Hivatkozás(sor):
  j:=1
  Ciklus amíg j<hossz(sor)
    Ciklus amíg j<hossz(sor) és sor(j)≠'&'
      j:=j+1
    Ciklus vége
    Ha j<hossz(sor)
       akkor k:=j; j:=j+1; s:=''
             Ciklus amíg sor(j)≠'&'
               s:=s+sor(j); j:=j+1
             Ciklus vége
             sor:=sor(1..k-1)+sor(j+1..hossz(sor))
             k:=1; Nagyravált(s)
             Ciklus amíg k≤névdb és név(k)≠s
               k:=k+1
             Ciklus vége
             Ha k>névdb akkor névdb:=névdb+1; név(névdb):=s
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

```
Szóeleje(i):
  Ha i=1 akkor Szóeleje:=igaz
      különben Szóeleje:=miben(i-1)=' '
Függvény vége.
```

```
Szóvége(i):
  Ha i=hossz(miben) akkor Szóvége:=igaz
  különben Szóvége:=miben(i+1)∈(' ','.',',',';',':','!','?')
Függvény vége.
```

Értékelési szempontok

A. Tud keresni az aktuális állományban	2 pont
B. Tud keresni a hivatkozás után is	1 pont
C. Tud tovább keresni csatolt állományban	2 pont
D. Tud keresni a csatolthoz csatoltban is	2 pont
E. Ugyanazt a csatolt állományt nem nézi meg még egyszer	2 pont
F. Nem zavarja a rekurzív hivatkozási lánc	2 pont
G. Csak teljes szót keres	2 pont
H. Szóhatárolóként alkalmazhatók írásjelek	2 pont
I. Hivatkozás a keresett karaktersorozat közepén is lehet	2 pont
J. Tud keresni a kis- és nagybetűk megkülönböztetése nélkül	2 pont
K. Nem ír ki olyan állománynevet, amelyben nincs benne a keresett	1 pont
L. Tud több keresést	1 pont
M. Nagy rendszerben is tud keresni	4 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Terv (50 pont)

Az A..D és az E..F részfeladatok megoldása összefügg, ezért egy-egy eljárásban megoldhatók.

Az `Idő` tömbbe olvassuk be a végrehajtási időket, majd építsük fel a munkák gráfját, külön a belépő élekre (Be(i) – az i-edik pontba belépő élek száma, GBe – a belépő élek, Ki(i) – az i-edik pontból kilépő élek száma, GKi – a kilépő élek):

```
Ki:=(0,...,0); Be:=(0,...,0)
Ciklus i=1-től M-ig
  Olvas(BeF,p,q)
  Ki(p):=Ki(p)+1; Be(q):=Be(q)+1
  GKi(p,Ki(p)):=q; GBe(q,Be(q)):=p
Ciklus vége
```

Első lépésként fűzzük listába azon pontokat, amelyekbe nem vezet bemenő él!

```
Megold1_4:
  MaxM:=0; Kezd1(0):=0; SFej:=0; Sorsz:=0
  Ciklus i=1-től N-ig
    L(i):=Be(i); Ős(i):=0
    Ha L(i)=0 akkor L(i):=SFej; SFej:=i
  Ciklus vége
```

Ezután járjuk be a kigyűjtött bemenő él nélküli pontokat és határozzuk meg a legkorábbi lehetséges végidejüket! Ehhez meg kell nézni, hogy a bevezető éleiken levő pontok közül melyiknek a legnagyobb a befejezési ideje és azt meg kell növelni az adott pont végrehajtási idejével! Ezután azon pontokat, amelyekbe már nem vezet több él, fel kell venni a feldolgozandók listájába!

```
Ciklus amíg SFej≠0
  p:=SFej; SFej:=L(SFej); Sorsz:=Sorsz+1; Sor(Sorsz):=p
  maxq:=GBe(p,1); max:=Kezd1(maxq)
  Ciklus i=2-től Be(p)-ig
    q:=GBe(p,i)
    Ha Kezd1(q)>max akkor max:=Kezd1(q); maxq:=q
  Ciklus vége
  Kezd1(p):=max+Idő(p); Ős(p):=maxq
  Ha Kezd1(MaxM)<Kezd1(p) akkor MaxM:=p
  Ciklus i=1-től Ki(p)-ig
    q:=GKi(p,i); L(q):=L(q)-1
    Ha L(q)=0 akkor L(q):=SFej; SFej:=q
  Ciklus vége
Ciklus vége
```

Ezután az A részfeladat megoldása annak a pontnak (munkának) a legkésőbbi befejezési ideje, amelyikre `Kezd1` maximális:

```
Mego1:=Kezd1(MaxM)
```

A B részfeladat megoldásához a `Kezd1` tömb elemeiből ki kell vonni a végrehajtási időket:

```
Ciklus i=1-től N-ig
  Kezd1(i):=Kezd1(i)-Idő(i)+1
Ciklus vége
```

A D részfeladat megoldása, a kritikus út azon pontokból áll, amerre a fenti bejárás során haladtunk (az `Ős` tömbben jegyeztük minden pontról, hogy mely pontból jöttünk oda):

```
KrUtVég:=0; p:=MaxM
Ciklus amíg p>0
  KrUtVég:=KrUtVég+1; KritUt(KrUtVég):=p; p:=Ős(p)
Ciklus vége
```

A C részfeladat megoldása – legkésőbbi kezdési idő – hasonló az A részfeladat megoldáshoz, csak hátulról visszafelé kell menni a pontokon (Kezd2 tömb):

```
Ciklus i=N-től 1-ig -1-esével
  p:=Sor(i); Vég:=Mego1
  Ciklus j=1-től Ki(p)-ig
    q:=GKi(p,j)
    Ha Kezd2(q)-Idő(q)<Vég akkor Vég:=Kezd2(q)-Idő(q)
  Ciklus vége
  Kezd2(p):=Vég
Ciklus vége
Ciklus i=1-től N-ig
  Kezd2(i):=Kezd2(i)-Idő(i)+1
Ciklus vége
Eljárás vége.
```

Az E részfeladat megoldásához a pontokat (munkák) a legkorábbi kezdési idejük szerinti sorrendbe rendezzük. Minden eddig megbízott vállalkozóról tároljuk, hogy mikortól kezdve szabad (azaz mikortól kaphat újabb munkát). Ha egy munka megkezdéséhez nincs az eddigiek között szabad vállalkozó, akkor újat kell megbízni. Ha van több szabad vállalkozó is, akkor azt kell választani, aki a

legkevesebbet kényszerül várakozni az előző munkája óta (ezen várakozási idők összege az F részfeladat megoldása):

```
Megold5_6:
  Ciklus i=1-től N-ig
    S(i):=i
  Ciklus vége
  Rendez; Mego5:=0; Mego6:=0
  Ciklus i=1-től N-ig
    j:=S(i); k:=0; unat:=+∞
    Ciklus l=1-től Mego5-ig
      Ha Szabad(l)≤Kezd1(j) és Kezd1(j)-Szabad(l)<unat
         akkor unat:=Kezd1(j)-Szabad(l); k:=l
    Ciklus vége
    Ha k=0 akkor Mego5:=Mego5+1
                 Szabad(Mego5):=Kezd1(j)+Idő(j)
        különben Mego6:=Mego6+Kezd1(j)-Szabad(k)
                 Szabad(k):=Kezd1(j)+Idő(j)
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A. részfeladat	5 pont (1 pont/teszt)
B. részfeladat	10 pont (2 pont/teszt)
C. részfeladat	10 pont (2 pont/teszt)
D. részfeladat	10 pont (2 pont/teszt)
E. részfeladat	10 pont (2 pont/teszt)
F. részfeladat	5 pont (1 pont/teszt)

2. feladat: Karaktersorozat keresése a Weben (25 pont)

A feladat és így a megoldása is azonos a kilencedik-tizedik osztályosok 2. feladatával.

Értékelési szempontok

A. Tud keresni az aktuális állományban	2 pont
B. Tud keresni a hivatkozás után is	1 pont
C. Tud tovább keresni csatolt állományban	2 pont
D. Tud keresni a csatolthoz csatoltban is	2 pont
E. Ugyanazt a csatolt állományt nem nézi meg még egyszer	2 pont
F. Nem zavarja a rekurzív hivatkozási lánc	2 pont
G. Csak teljes szót keres	2 pont
H. Szóhatárolóként alkalmazhatók írásjelek	2 pont
I. Hivatkozás a keresett karaktersorozat közepén is lehet	2 pont
J. Tud keresni a kis- és nagybetűk megkülönböztetése nélkül	2 pont
K. Nem ír ki olyan állománynevet, amelyben nincs benne a keresett	1 pont
L. Tud több keresést	1 pont
M. Nagy rendszerben is tud keresni	4 pont

3. feladat: Kommandó (25 pont)

A feladat egy speciális gráfban legrövidebb út kereséséről szól. A gráf pontjai legyenek az országok, élek pedig azon pontok között legyenek, amely országok szomszédosak egymással! Ha egy él két végpontja azonos színű, akkor a hossza legyen 0, ha különböző színű, akkor pedig 1!

A gráf felépítése a következő lehet (`szín(i)` legyen az i-edik ország színe, `db(i)` az i-edik ország szomszédai száma, `él(i,j)` pedig az i-edik ország j-edik szomszédjának sorszáma):

```
G:=((+∞,...),...,(...+∞))
Ciklus i=1-től N-ig
  Ciklus j=1-től db(i)-ig
    Ha szín(i)=szín(él(i,j)) akkor G(i,j):=0; G(j,i):=0
                             különben G(i,j):=1; G(j,i):=1
  Ciklus vége
Ciklus vége
```

Ezután nincs más teendő, mint alkalmazni a gráfok szélességi bejárásának módosított algoritmusát (a prioritási sor a `Táv` vektor alapján épül)!

```
Táv:=(+∞,...,+∞); Táv(indul):=0
PrSorba(indul); Honnan(indul):=indul; x:=indul
Ciklus amíg nem üres(PRSOR) és x≠cél
  PrSorból(x)
  Ciklus i=1-től N-ig
    Ha G(x,i)<+∞ és Táv(i)≥Táx(x)+G(x,i)
       akkor Honnan(i):=x; Táv(i):=Táv(x)+G(x,i); PrSorba(i)
  Ciklus vége
Ciklus vége
```

A célpontból visszafelé bejárva az utat megkaphatjuk mind a színek, mind az útvonalon fekvő országok halmazát.

```
SZ:={szín(cél)}; ORSZ:={cél}; x:=cél
Ciklus amíg x≠indul
  x:=Honnan(x); SZ:=SZ∪{szín(x)}; ORSZ:=ORSZ∪{x}
Ciklus vége
```

Értékelési szempontok

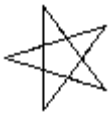
A. Egyetlen út esetén megadja a színkódokat	3 pont
B. Több út, de van egyetlen trikós esetén megadja az utat	4 pont
C. Több út van, legrövidebb esetén megadja a megoldást	3+3 pont
D. Több út van, legrövidebb, többféle trikó esetén megadja a megoldást	3+4 pont
E. Nagy adathalmaz	5 pont

1999. Első forduló

Ötödik-nyolcadik osztályosok

1. feladat: Mit rajzol (25 pont)

A.	5 ágú csillagot rajzol	4 pont
	36 fok a csillag hegyeinél levő belső szög	3 pont
	108 fok a csillag ágai közötti külső szög	3 pont
B.	11 ágú csillagot rajzol	4 pont
	30 fok a csillag hegyeinél levő belső szög	3 pont
	kb. 62.7 fok (pontosan: 180–1290/11) a csillag ágai közötti külső szög	3 pont
C.	A csillag ágainak belsejét is berajzolja	5 pont

A. ábra: 	B. ábra: 	C. ábra: 

2. feladat: Robot (25 pont)

A.	`Margitsziget`	3 pont
B.	B1 jó	2 pont
	B2 rossz, Pestről az 5-ös híd nem érhető el	1+2 pont
	B3 rossz, Budáról a 6-os híd nem érhető el	1+2 pont
	B4 rossz, a Margitszigetről az 1-es híd nem érhető el	1+2 pont
C.	Az 5,5, illetve a 6,6 párok elhagyhatók	2 pont
	Az 5,6, illetve a 6,5 párok 1..4-esre cserélhetők	2 pont
	Az x,y párok (ahol x és y 1..4 lehet) elhagyhatók	2 pont
	Az x,6 párok (ahol x 1..4 lehet) 5-re cserélhetők	2 pont

Megjegyzés: További egyszerűsítő szabályok nem adnak újabb rövidítési lehetőséget.

D.	5 esetén a végállapot a Margitsziget	1 pont
	Üres sorozat esetén a végállapot Buda	1 pont
	Egyetlen 1..4 közötti érték esetén a végállapot Pest	1 pont

3. feladat: Cserélgetés (25 pont)

A.	`M` az `A`-beli értékek maximuma	3 pont
	`D` azoknak az értékeknek, az ún. ideiglenes maximumoknak a száma, amelyek a maximumkeresés közben átmenetileg maximálisak voltak	4 pont
	`Y` az ideiglenes maximumokat tartalmazza	3 pont
B.	Az `A` vektorba minden elem helyébe az addigi ideiglenes maximum kerül	5 pont
C.	Az `A` vektor összes eleme egyforma lesz, ha az első eleme a maximum	5 pont
D.	Az `A` vektor nem változik, ha eredetileg már monoton növekedő volt	5 pont

4. feladat: Véletlen (25 pont)

A. Az `A` tömb `0` és `M` közé eső egész számokat tartalmazhat — 3 pont
Az `A` tömb elemei között mindig van 0 értékű — 3 pont
A számok között nincs két egyforma — 3 pont
A számok sorrendje véletlenszerű — 3 pont

B. A ciklus *utolsó* öt lépésében a véletlen értékek rendre `2, 3, 4, 5, 6` — 4 pont
Az eljárás utolsó utasításában a véletlen érték `1` — 1 pont

C. A ciklus *első* hat lépésében a véletlen értékek mind `1`-esek — 4 pont
A többiben `1` vagy legalább `6` — 3 pont
Az eljárás utolsó utasításában a véletlen érték `1` — 1 pont

Kilencedik-tizedik osztályosok

1. feladat: Halmazok metszete (16 pont)

A. Ha az `A` vektor a `B` vektor első `N` elemét tartalmazza tetszőleges sorrendben — 4 pont
Részmegoldás: ha megad egy példát, Pl. `B(1)=A(1) ... B(N)=A(N)` 1 pont
`N*(N–1)/2` lépésben (a belső ciklus lépésszáma az összes 0 és `N`-1 közötti érték) — 4 pont

B. Ha az `A` és a `B` vektornak nincs közös eleme — 4 pont
`N*M` lépésben (a külső ciklust `N`, a belsőt `M` lépésben) — 4 pont

2. feladat: Festegetés (15 pont)

A.

36	35	34	33	32	31
17	16	15	14	13	30
18	5	4	3	12	29
19	6	1	2	11	28
20	7	8	9	10	27
21	22	23	24	25	26

5 pont

B.

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

5 pont

C.

22	21	20	19	18	17
23	6	5	4		16
24	7		3		15
25	8	1	2		14
26	9	10	11	12	13
27					

5 pont

3. feladat: Egyesítés (20 pont)

A. `A(N)=B(M)` — 4 pont
`A` minden eleme különböző — 3 pont
`B` minden eleme különböző — 3 pont

B. Ha `A(N)<B(M)`, akkor az `A(N)`-nél nagyobb `B`-beliek nem kerülnek át `C`-be — 3 pont
Ha `A(N)>B(M)`, akkor a `B(M)`-nél nagyobb `A`-beliek nem kerülnek át `C`-be — 3 pont
Ha egy elem `A`-ban többször megvan, akkor az többször kerül be `C`-be — 2 pont
Ha egy elem `B`-ben többször megvan, akkor az többször kerül be `C`-be — 2 pont

4. feladat: Kiskapus sor (25 pont)

A1. Igen, `SORBA, SORBA, ÁT, SORBÓL, ÁT, SORBÓL, ÁT` — 2+3 pont

A2. Nem, 2,4-ig állítható elő — 2+2 pont

A3. Nem, 5,1,2-ig állítható elő — 2+2 pont

A4. Igen, `SORBA, ÁT, ÁT, SORBÓL, SORBA, ÁT, SORBÓL` — 2+3 pont

B. Egy jel akkor kerülhet az eredeti helyénél előbbre, ha az általa megelőzöttek sorrendjének nem kell változnia (ugyanis azoknak a sorban kell várakozniuk) 7 pont

Másképp megfogalmazva: ha i<j<k, akkor a k,j,i sorrend nem lehetséges.

5. feladat: Megszakítások (24 pont)

A. `Megszakításkérés` nem állítja át `AKT` értékét, ha `P` nagyobb volt nála 4 pont

`Ha P>AKT akkor AKT:=P` a `Megszakításkérés` eljárás végén 4 pont

B. `Kiszolgálás` nem veszi ki a sorból a már kiszolgált megszakítást 4 pont

`Ha Első(AKT).IDŐ=10 akkor Sorból(AKT)` a `Kiszolgálás` eljárás végén 4 pont

C. `Kiszolgálás` nem állítja `VAN`-t hamisra, ha nincs több megszakításkérés 4 pont

`VAN:=(AKT=1 és Üres(AKT))` a `Kiszolgálás` eljárás végén 4 pont

Részmegoldás: ha rájön, hogy `Kiszolgálás` a ciklusból `AKT=0`-val kilépve hibásan hajtja végre az eljárás hátralevő részét, és erre megoldást ad 2+2 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Időszerű (15 pont)

A1. Hibás, ha `A` és `B` utolsó rekordjának a kulcsa nem egyenlő egymással 3 pont

A2. Ha közülük `A` utolsó rekordja a kisebb kulcsú, akkor `B`-ben maradnak olyan új rekordok, amelyek nem kerülnek be `C`-be 2 pont

Ha közülük `A` utolsó rekordja a nagyobb kulcsú, akkor `A`-ban maradnak olyan rekordok, amelyek nem kerülnek be `C`-be 2 pont

A3. A ciklus lefutása után az `A`-ban maradt rekordokat át kell másoltatni `C`-be 2 pont

A ciklus lefutása után a `B`-ben maradt beszúró parancsokat végre kell hajtatni 2 pont

B. A `B` új rekordot akar beszúrni, de az `A`-ban van ilyen kulcsú rekord 2 pont

A `B` rekordot akar törölni, de az `A`-ban nincs ilyen kulcsú rekord 1 pont

A `B` rekordot akar módosítani, de az `A`-ban nincs ilyen kulcsú rekord 1 pont

2. feladat: Családfa (15 pont)

A1. Az eredmény `f` gyerekeinek a száma 4 pont

A2. Az eredmény a gyerektelenek száma a családfában 6 pont

B. A helyes megoldás: 5 pont

```
LegkisebbGyerek(f):
  f:=Balra(f)
  Ciklus
    g:=f; f:=Jobbra(f)
  amíg Siker
  Ciklus vége
  LegkisebbGyerek:=g
Eljárás vége.
```

3. feladat: Logikusan (17 pont)

A. `MilyenA(X)` teljesül, ha van `X`-nél fiatalabb 3 pont

`Éva`, `András`, `Anna` is lehet a megoldás 1+1+1 pont

B. `MilyenB(X)` teljesül, ha `X` a legidősebb 3 pont

`András` a megoldás 1 pont

`Valami(E)` teljesül, ha van valaki, aki az `E` év előtt született 2 pont

C. `MilyenC(X)` teljesül, ha `X` és `Y` születési évének különbsége minimális 2 pont

`Éva` és `Anna` a megoldás 1 pont

`Ilyen(G)` teljesül, ha vannak, akiknek a születési éve között `G`-nél kisebb a különbség 2 pont

4. feladat: Kitalálós (15 pont)

A. `Mitcsinál(A,B)` értéke `A+B` 4 pont

`Találdki(A,B)` értéke `A-B` 4 pont

B. `B`-ben a bitenkénti átvitelek aktuális értéke van 4 pont

C. Maximum 16-szor hajtják végre a ciklus magját 3 pont

5. feladat: Megszakítások (21 pont)

A. `Megszakításkérés` nem állítja át `AKT` értékét, ha `P` nagyobb volt nála 4 pont

`Ha P>AKT akkor AKT:=P` a `Megszakításkérés` eljárás végén 3 pont

B. `Kiszolgálás` nem veszi ki a sorból az éppen kiszolgált megszakítást 4 pont

`Ha Első(AKT).IDŐ=10 akkor Sorból(AKT)` a `Kiszolgálás` eljárás végén 3 pont

C. `Kiszolgálás` nem állítja `VAN`-t hamisra, ha nincs több megszakításkérés 4 pont

`VAN:=(AKT=1 és Üres(AKT))` a `Kiszolgálás` eljárás végén 3 pont

Részmegoldás: ha rájön, hogy `Kiszolgálás` a ciklusból `AKT`=0-val kilépve hibásan hajtja végre az eljárás hátralevő részét, és erre megoldást ad 2+2 pont

6. feladat: Gyorskereső (17 pont)

A. `SH(I)`=az `I`. karakter jobbról hányadik helyen fordul elő először a mintában 3 pont

Ha nem fordul elő a mintában, akkor `SH(I)=M+1` 2 pont

B. A minta *egyetlen karaktere sem* fordul elő a szövegben 3 pont

`N div M` 3 pont

C. A minta *utolsó karakterét kivéve,* a szöveg és a minta *csupa egyforma* karakterből áll 3 pont

`N-M+1` 3 pont

1999. Második forduló

Ötödik-nyolcadik osztályosok

1. feladat: Szimmetrikusan (15 pont)

Elkezdjük elölről és hátulról is nézni a karaktereket, s ha különböznek, akkor kiírjuk a hely sorszámát (elölről). A `jó` változó igaz marad, ha a virágszín sorozat tökéletes.

```
jó:=igaz
Ciklus i=1-től N div 2-ig
  Ha t(i)≠t(N-i+1) akkor Ki: i; jó:=hamis
Ciklus vége
Ha jó akkor Ki: 'Tökéletes'
```

Értékelési szempontok

A. Szimmetrikus eset, páros számú elemmel 3 pont

B. Szimmetrikus eset, páratlan számú elemmel 3 pont

C. Az első helyen nem szimmetrikus 3 pont

D. Több helyen nem szimmetrikus 3 pont

E. Páros számú elem esetén a közepén nem szimmetrikus 3 pont

2. feladat: Naptár és óra (29 pont)

Tárolnunk kell konstans tömbökben a hónapok nevét és napszámát, valamint a hét napjai nevét:

```
hónév=('január','február','március','április','május',
'június','július','augusztus','szeptember','október',
'november','december')

napszám=(31,28,31,30,31,30,31,31,30,31,30,31)

napnév=('hétfő','kedd','szerda','csütörtök','péntek',
'szombat','vasárnap')
```

A február napjai száma szökőévekben eggyel nagyobb, így még ezzel is kell foglalkoznunk:

```
szökőév(i):
  szökőév:=(i mod 400)=0 vagy (i mod 4)=0 és (i mod 100)≠0
Függvény vége.
```

Ismételni kell a következőket: először a másodpercet növeljük, ezután ha kell, akkor a percet, majd ha kell, akkor az órát, … és így tovább:

```
Ki: év,hónév[hó],nap,napnév[napsor],óra,perc,mp
Ha mp<59 akkor mp:=mp+1 különben
   mp:=0
   Ha perc<59 akkor perc:=perc+1 különben
      perc:=0
      Ha óra<23 akkor óra:=óra+1 különben
         óra:=0
         Ha napsor=7 akkor napsor:=1
                  különben napsor:=napsor+1
         Ha szökőév(év) és (hó≠2 vagy nap<29)
            vagy (nap<napszám(hó)) akkor nap:=nap+1
         különben nap:=1
                  Ha hó<12 akkor hó:=hó+1
                        különben hó:=1; év:=év+1
      Elágazás vége
   Elágazás vége
Elágazás vége
```

Értékelési szempontok

1. teszt: `1998. január 23. péntek, 23 óra 58 perc 58 másodperc`

Jól nő a másodperc 2 pont

Jól vált percet 2 pont

Jól vált órát 2 pont

Jól vált napot 2 pont

Jól váltja a hét napjait (péntek → szombat) 3 pont

2. teszt: `1998. január 31. szombat, 23 óra 59 perc 58 másodperc`

Jól váltja a hét napjait (szombat → vasárnap) 2 pont

Jól vált 31 napos hónapot 3 pont

3. teszt: `1998. november 30. hétfő, 23 óra 59 perc 58 másodperc`

Jól vált 30 napos hónapot 3 pont

4. teszt: `1998. február 28. szombat, 23 óra 59 perc 58 másodperc`

Jól vált 28 napos hónapot 2 pont

5. teszt: `1998. december 31. csütörtök, 23 óra 59 perc 58 másodperc`

Jól vált évet 2 pont

6. teszt: `1996. február 28. szombat, 23 óra 59 perc 58 másodperc`

Jól kezeli a szökőéveket (február 29 van) 2 pont

7. teszt: `1900. február 28. szombat, 23 óra 59 perc 58 másodperc`

1900 nem szökőév (február 29 nincs) 2 pont

8. teszt: `2000. február 28. szombat, 23 óra 59 perc 58 másodperc`

2000 szökőév (február 29 van) 2 pont

3. feladat: Mozijegyek (31 pont)

Először számoljuk ki az iskolalétszámot:

```
iskolalétszám:=0
Ciklus i=1-től M-ig
  iskolalétszám:=iskolalétszám+létszám(i)
Ciklus vége
```

Ezután számoljuk ki, hogy létszámarányosan mennyi jegy járna az egyes osztályoknak, majd rendezzük sorba őket a törtrész szerint!

```
Ciklus i=1-től M-ig
  db(i):=n*létszám(i)/iskolalétszám; sor(i):=i
Ciklus vége
Ciklus i=1-től M-1-if
  min:=i
  Ciklus j=i+1-től M-ig
    Ha törtrész(db(j))<törtrész(db(min)) akkor min:=j
  Ciklus vége
  k:=sor(i); sor(i):=sor(min); sor(min):=k
  x:=db(i); db(i):=db(min); db(min):=x
Ciklus vége
```

Számoljuk meg, hogy az egészrészek összegét (az eddig kiosztott jegyek számát), és a maradék jegyeket a törtrész szerint csökkenő sorrendben adjuk az egyes osztályoknak!

```
jegy:=0
Ciklus i=1-től M-ig
  jegy:=jegy+egészrész(db(i))
Ciklus vége
Ciklus i=M-től 1-ig -1-esével
  Ha jegy<N akkor db(i):=egészrész(db(i))+1; jegy:=jegy+1
          különben db(i):=egészrész(db(i))
Ciklus vége
```

Értékelési szempontok

A. Minden hányados egész 4 pont

B. Két hányados tört, egyformák 4 pont

C. Két hányados tört, nem egyformák 4 pont

D. Minden hányados tört, egyformák 4 pont

E. Vannak tört hányadosok, mind különböző, a felét lefelé kell kerekíteni 5 pont

F. Vannak tört hányadosok, mind különböző, egyet kell lefelé kerekíteni 5 pont

G. Vannak tört hányadosok, mind különböző, egyet kell felfelé kerekíteni 5 pont

Kilencedik-tizedik osztályosok

1. feladat: OKTV (13 pont)

A feladat egy mátrix oszlopai rendezése az oszlop, mint sorozat szerint lexikografikusan. A megoldás gyorsrendezéssel:

```
Rendez(Bal,Jobb):
  Ciklus amíg Bal<Jobb
    f:=Feloszt(Bal,Jobb); Rendez(Bal,f); Bal:=f+1
  Ciklus vége
Eljárás vége.
```

```
Feloszt(Bal,Jobb):
  Fi:=S((Bal+Jobb) Div 2); i:=Bal; j:=Jobb
  Ciklus amíg i<j
    Ciklus amíg Lexiko(S(i),Fi)
      i:=i+1
    Ciklus vége
    Ciklus amíg Lexiko(Fi,S(j))
      j:=j-1
    Ciklus vége
    Ha i<j akkor Csere(S(i),S(j)); i:=i+1; j:=j-1
  Ciklus vége
  Feloszt:=i
Függvény vége.
```

```
Lexiko(i,j):
  k:=1
  Ciklus amíg k≤M és A(i,k)=A(j,k)
    k:=k+1
  Ciklus vége
  Lexiko:=(k≤M) és A(i,k)>A(j,k)
Függvény vége.
```

Értékelési szempontok

A. Az 1. forduló után már nincs holtverseny 1 pont

B. Az első fordulóban a győztes eldőlt, a további helyeken holtverseny van 1 pont

C. Az első fordulóban a győztes és az utolsó eldőlt, a további helyeken több holtverseny van 1 pont

D. A győztes is újra indul, de igen rossz eredménnyel 1 pont

Az utolsó is újra indul, de igen jó eredménnyel 1 pont

E. Több forduló kell, mindig csökken a holtversenyesek száma 1 pont

F. Több forduló kell, az utolsóig ugyanazok vannak holtversenyben 1 pont

G. Későbbi holtversenyesek több pontot, szereznek, mint a korábbi holtversenyesek 1 pont

H. Nagyon sok adat 2+2 pont

I. Legtöbb adat (N=1000) 2 pont

2. feladat: Család (24 pont)

A feladatban egy családfáról teszünk fel különböző kérdéseket. Az A részfeladat megoldásához meg kell vizsgálni, hogy a keresett személynek kik a testvérei!

```
A_Feladat:
  adb:=0
  Ciklus x=1-től M-ig
    Ha Testvér(Sz,x) akkor adb:=adb+1; A(adb):=x
  Ciklus vége
Eljárás vége.

Testvér(x,y):
  Testvér:=x≠y és Anya(x)>0 és Apa(x)>0 és
           Anya(x)=Anya(y) és Apa(x)=Apa(y)
Függvény vége.
```

A B részfeladat ugyanezt teszi a féltestvérekkel:

```
B_Feladat:
  bdb:=0
  Ciklus x=1-től M-ig
    Ha Féltestvér(Sz,x) akkor bdb:=bdb+1; B(bdb):=x
  Ciklus vége
Eljárás vége.

Féltestvér(x,y):
  Féltestvér:=x≠y és (Anya(x)>0 és Anya(x)=Anya(y) kizáróvagy
                      Apa(x)>0 és Apa(x)=Apa(y))
Függvény vége.
```

A C részfeladatban az apai ágon kell menni felfelé, amíg csak lehet:

```
C_Feladat:
  cdb:=0; x:=Apa(Sz)
  Ciklus amíg x>0
    cdb:=cdb+1; C(cdb):=x; x:=Apa(x)
Ciklus vége
Eljárás vége.
```

Azt keressük, akinek a keresett személlyel van közös nagyszülője, azaz a szülei közül az egyik legalább féltestvére a másik szülei egyikének:

```
D_Feladat:
  ddb:=0
  Ciklus x=1-től M-ig
    Ha x≠Sz és
       ((Anya(Sz)>0 és Anya(x)>0 és van(Anya(Sz),Anya(x)) vagy
        (Anya(Sz)>0 és Apa(x)>0 és van(Anya(Sz),Apa(x)) vagy
        (Apa(Sz)>0 és Anya(x)>0 és van(Apa(Sz),Anya(x)) vagy
        (Apa(Sz)>0 és Apa(x)>0 és van(Apa(Sz),Apa(x)))
    akkor ddb:=ddb+1; D(ddb):=x
  Ciklus vége
Eljárás vége.

van(x,y):
  van:=x≠y és (Anya(x)>0 és Anya(x)=Anya(y) vagy
               Apa(x)>0 és Apa(x)=Apa(y))
Függvény vége.
```

Értékelési szempontok

A. CSALAD1

Nincs testvér	1 pont
Nincs féltestvér	1 pont
Csak az apja van apai felmenőként	2 pont
Nincs unokatestvér	2 pont

B. CSALAD2

1 testvér van	1 pont
Apai féltestvér van	1 pont
Sok férfiági felmenő van	2 pont
Van 1 unokatestvér	2 pont

C. CSALAD3

2 testvér van	1 pont
Anyai féltestvér van	1 pont
Férfiági felmenők	2 pont
Mind a 4 nagyszülőn keresztül van unokatestvér	2 pont

D. CSALAD4

Testvér	1 pont
Mindkét féle féltestvér van	1 pont
Férfiági felmenők	2 pont
Mind a 4 nagyszülő közös (az apák és az anyák is testvérek)	2 pont

3. feladat: Szójáték (18 pont)

Tekintsük azt a gráfot, amelynek pontjai a szótár szavai, és P→Q csak akkor él, ha a P szóból a megadott szabály alkalmazásával kapható a Q szó. Az 1. részfeladat megoldása tehát a megadott P szóból a Q szóba vezető legrövidebb út hossza a gráfban, ha létezik. A 2. részfeladat pedig az összes P-ből elérhető pontok meghatározását jelenti.

A megoldást szélességi bejárással végezzük.

```
Bejárás:
  Lépés:=(0,...,0); Pi:=Szó(P); Qi:=Szó(Q); OK:=hamis
  Lépés(Pi):=1; Sorba(S,Pi)
  Ciklus amíg nem Üres(S)
    Sorból(S,i)
    Ciklus j=1-től N-ig
      Ha Lépés(j)=0 és Jóe(i,j)
         akkor Lépés(j):=Lépés(i)+1; OK:=OK vagy j=Qi
               Sorba(S,j)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A függvény értéke legyen igaz, ha az i-edik szóból megkapható a j-edik! Ha a B szabályt alkalmazzuk (az AvB változó igaz értékű), akkor az eggyel hosszabb szót is el kell fogadni!

```
Jóe(i,j):
  li:=hossz(Szótár(i)); lj:=hossz(Szótár(j)); k:=1
  Ciklus amíg Szótár(i,k)=Szótár(j,k)
    k:=k+1
  Ciklus vége
  vég:=k; k:=k+1
  Ciklus amíg k≤li és k≤lj és Szótár(i,k)=Szótár(j,k)
    k:=k+1
  Ciklus vége
  Jóe:=(li=lj és k=li+1 vagy AvB és vég=lj és lj=li+1)
Függvény vége.
```

A függvény megadja egy szó szótárbeli indexét:

```
Szó(S:):
  i:=1
  Ciklus amíg Szótár(i)≠S
    i:=i+1
  Ciklus vége
  Szó:=i
Függvény vége.
```

Értékelési szempontok

A szabály:

A. A-ból B egy lépéssel előállítható	1+1 pont
B. A-ból B több lépéssel állítható elő	2+1 pont
C. A-ból B nem állítható elő	1+1 pont
D. A-ból semmi sem állítható elő	1+1 pont

B szabály újdonságai:

E. B rövidebb A-nál, nem állítható elő	2+1 pont
F. B a végére írással állítható elő	2+1 pont
G. több szabályt kell felváltva alkalmazni	2+1 pont

4. feladat: Kockák (20 pont)

Legyen `Kocka(i)` a legmagasabb torony magassága, ahol az i-edik kocka van felül:

$$Kocka(i) = \max\begin{cases} Kocka(1)+1 & ha \quad M_i \geq M_1 \\ Kocka(2)+1 & ha \quad M_i \geq M_2 \\ \quad ... & \\ 1 & egyébként \end{cases}, \qquad Kocka(1) = 1$$

Először rendezzük a kockákat súly szerint nemcsökkenő sorrendbe! A képlet alapján kiszámolva a `Kocka` tömböt, a megoldás a `Kocka` tömb maximuma:

```
Kocka(1):=1; Mire(1):=0
Ciklus i=2-től N-ig
  Kocka(i):=1; Mire(i):=0
  Ciklus j=1-től i-1-ig
    Ha M(i)≥M(j) és Kocka(i)<Kocka(j)+1
       akkor Kocka(i):=Kocka(j)+1; Mire(i):=j
  Ciklus vége
Ciklus vége
max:=1
Ciklus i=2-től N-ig
  Ha Kocka(i)>Kocka(max) akkor max:=i
Cikkus vége
```

A megoldás a `Mire` tömb alapján rekurzívan visszafelé írható ki:

```
Kiírás(i,db):
  ha i=0 akkor Ki: db
      különben Kiírás(Mire(i),db+1); Ki: hossz(i); súly(i)
Eljárás vége.
```

Értékelési szempontok

A. Minden kocka felhasználható	2 pont
B. Csak egyetlen kocka használható (a méret csökkenésével a súly nő)	2 pont
C. Minden második használható, kevés adattal	2 pont
D. Minden kocka felhasználható, sorrend változik (N=101)	2 pont
E. Minden második kocka használható sok adattal (N=200)	2 pont
F. Minden kocka felhasználható, sorrend változik (N=991)	2 pont
G. Közepes fa (N=210, 20 kocka lehet 20 1-től 1 1-ig)	2 pont
H. Nagy fa (N=990, 44 kocka lehet 44 1-től 1 1-ig)	2 pont
I. Közepes rács-gráf (N=500=5*100, 104 kocka lehet 5 100-tól 5 1-ig, majd 4 1, 3 1, 2 1, 1 1)	2 pont
J. Nagy rács-gráf (N=1000=50*20, 69 kocka lehet 50 20-tól 50 1-ig, majd 49 1-től 1 1-ig)	2 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Kockák (15 pont)

A feladat, és így a megoldása is, azonos a kilencedik-tizedik osztályosok negyedik feladatával.

Értékelési szempontok

A. Minden kocka felhasználható	1 pont
B. Csak egyetlen kocka használható (a méret csökkenésével a súly nő)	1 pont
C. Minden második használható, kevés adattal	1 pont
D. Minden kocka felhasználható, sorrend változik (N=101)	2 pont
E. Minden második kocka használható sok adattal (N=200)	2 pont
F. Minden kocka felhasználható, sorrend változik (N=991)	2 pont
G. Közepes fa (N=210, 20 kocka lehet 20 1-től 1 1-ig)	2 pont

H. Nagy fa (N=990, 44 kocka lehet 44 1-től 1 1-ig) 2 pont

I. Közepes rács-gráf (N=500=5*100, 104 kocka lehet 5 100-tól 5 1-ig, majd 4 1, 3 1, 2 1, 1 1) 2 pont

J. Nagy rács-gráf (N=1000=50*20, 69 kocka lehet 50 20-tól 50 1-ig, majd 49 1-től 1 1-ig) 2 pont

2. feladat: Barátságok (16 pont)

Mivel minden tanuló pontosan egy másikat jelöl meg, mint neki legszimpatikusabbat, így ezt a kapcsolatot egy F:1..N→1..N függvény írja le, F(x) az x-nek legszimpatikusabb tanuló sorszáma. A részfeladatok megoldásai kifejezhetők az F függvény gráfja alapján, ahol az irányított élek az x→F(x) párok. Jelölje Befok(x) azon y-ok számát, amelyre F(y)=x!

```
A_Feladat:
  adb:=0
  Ciklus x=1-től N-ig
    Ha BeFok(x)=0 akkor adb:=adb+1; A(adb):=Sz(x)
  Ciklus vége
Eljárás vége.
```

B. Azon x-F(x) párok, amelyekre x=F(F(x)) és Befok(x)=1, Befok(F(x))=1.

```
B_Feladat:
  bdb:=0
  Ciklus x=1-től N-ig
    Ha x<F(x) és F(F(x))=x és BeFok(x)=1 és BeFok(F(x))=1
       akkor bdb:=bdb+1; B(bdb):=Sz(x)
  Ciklus vége
Eljárás vége.
```

C. Azon x-ek, amelyekre Befok(x) maximális.

```
C_Feladat:
  Max:=0; cdb:=0
  Ciklus x=1-től N-ig
    Ha BeFok(x)>Max akkor Max:=BeFok(x)
  Ciklus vége
  Ciklus x=1-től N-ig
    Ha BeFok(x)=Max akkor cdb:=cdb+1; C(cdb):=Sz(x)
  Ciklus vége
Eljárás vége.
```

D. Nyilvánvaló, hogy minden x-re x és F(x) ugyanabban a csoportban kell, hogy legyen. Jelölje $F^k(x)$ az F függvény k-szori alkalmazását, azaz $F^1(x)=x$, $F^{k+1}(x)=F(F^k(x))$. Általánosan, az $x,F(x),...,F^k(x)$ sorozat minden elemének ugyanabban a csoportban kell lennie. Körön olyan $x,F(x),...,F^k(x)$ sorozatot értünk, ahol $F(F^k(x))=x$. Tehát a lehetséges csoportok maximális száma megegyezik a függvény gráfjában levő körök számával, hiszen az egy körben levőknek nyilván egy csoportban kell lenniük, továbbá bármely pontból indulva előbb-utóbb egy körbe jutok, így az érintett pontok is abban a csoportban vannak, mint azon kör elemei, ahova eljutottunk.

Így a következő algoritmus nyilvánvalóan helyes eredményt ad.

Az összes x pontra: Ha x még nincs csoportba osztva akkor képezzük az $x, F(x),..., F^k(x)=y$ sorozatot mindaddig, amíg

- a kapott y elem vagy megegyezik a sorozat egy korábbi tagjával,
- vagy y-t már beosztottuk a Csop(y) csoportba.

Az első esetben képezzünk egy új csoportot és a sorozat elemeit osszuk be ebbe a csoportba! A második esetben a sorozat elemeit osszuk be a Csop(y) csoportba!

```
D_Feladat:
  MaxCsop:=0
  Ciklus x=1-től N-ig
    Új(x):=igaz; Csop(x):=0
  Ciklus vége
  Ciklus x=1-től N-ig
    Ha Új(x)
       akkor x0:=x; Új(x):=hamis
             Ciklus amíg Új(F(x))
               x:=F(x); Új(x):=hamis
             Ciklus vége
             x1:=x
             Ha Csop(F(x1))=0
                akkor MaxCsop:=MaxCsop+1; Cs:=MaxCsop
             különben Cs:=Csop(F(x1))
             x:=x0
             Ciklus amíg x≠x1
               Csop(x):=Cs; x:=F(x)
             Ciklus vége
             Csop(x1):=Cs
  Ciklus vége
Eljárás vége.
```

Értékelési szempontok

A. Egyetlen gyűrű	1+1+1+1 pont
B. Egy szimpatikus pár, egy gyűrű és egy fa	1+1+1+1 pont
C. Nagy gráf	1+1+1+1 pont
D. Nagyon nagy gráf (N=1000), véletlen adatokkal	1+1+1+1 pont

3. feladat: Terv (14 pont)

A sík minden P(x,y) pontjához rendeljük hozzá azt a szöget, amelyet az y-tengely és az Origón és a P ponton áthaladó egyenes bezár!

Látható, hogy a T1(x1,y1,dx1,dy1) és T2(x2,y2,dx2,dy2) téglalapok csak akkor nem takarják egymást, ha T1 jobb alsó P1(x1+dx1,y1) sarkához tartozó szög kisebb, mint T2 bal felső P2(x2,y2+dy2) sarkához tartozó szög, vagy fordítva.

Rendezzük a téglalapokat a jobb alsó sarkukhoz tartozó szög szerint! Ebben a sorrendben mohó választással válogassuk be a téglalapokat!

Az 1. téglalapot válasszuk be; majd az összes többire, ha az nincs takarásban egyetlen már beválasztott téglalappal sem, akkor az i. téglalapot válasszuk be!

A „nincs takarásban egyetlen már beválasztott téglalappal sem" feltétel egyszerűen számítható, mert azt jelenti, hogy az utolsónak beválasztott téglalap jobb alsó sarka előbb van, mint az i-edik bal felső sarka.

Az algoritmus helyessége: Legyen A(1), A(2),...,A(K) optimális megoldás, és a téglalapok legyenek rendezve a fenti módon! Ekkor az A(1) téglalap biztosan kicserélhető a T(1) téglalappal, mert T(1) a rendezés szerint az összes közül a legkisebb szöget adja a jobb alsó sarkával, tehát a jobb alsó sarkához tartozó szög nem nagyobb, mint A(1) jobb alsó sarkához tartozó szög. Tehát T(1) sem takarhat egyetlen téglalapot sem az A(2),..., A(K) közül.

Indukciót alkalmazva belátható, hogy a mohó algoritmus optimális megoldást állít elő.

```
Szám:=1; Vég.x:=T(1).x+T(1).dx; Vég.y:=T(1).y; Be(1):=1
Ciklus i=2-től N-ig
  Ha Vég.x*(T(i).y+T(i).dy)<T(i).x)*Vég.y
     akkor Szám:=Szám+1; Be(Szám):=i
           Vég.x:=T(i).x+T(i).dx; Vég.y:=T(i).y
Ciklus vége
```

Értékelési szempontok

A. Több épület, nincs takarás	1 pont
B. Azonos bal sarkú épületek	1 pont
C. Egy mindent takar	1 pont
D. Előre haladó sorozat i-edik eleme részben takarja az i+1-edik elemet	1 pont
E. Vissza haladó sorozat i-edik eleme részben takarja az i+1-edik elemet	1 pont
F. A D. és az E. kombinációja	1 pont
G. Azonos méretű épületek egy sorban	1 pont
H. Nem azonos méretű épületek egy sorban	1 pont
I. Épületek körben vannak	1 pont
J. Véletlen adatok, N=100	1 pont
K. Véletlen adatok, N=1000	2 pont
L. Véletlen adatok, N=5000	2 pont

4. feladat: Kamion (15 pont)

A feltételekből következik, hogy az úthálózat alkotta gráf fa, aminek gyökere az 1. város, ahova szállítani kell. Definiáljuk minden v városra az alábbi értékeket:

- K(v): a v városon áthaladó kamionok száma;
- E(v): a v város gyökerű fában levő városokban termelt mennyiségek összege;
- M(v): a minimális kamion kapacitás, amivel a v gyökerű fában levő városokból a v-be történő szállítás elvégezhető.

A feladat megoldása nyilvánvalóan M(1).

Ha v-be nem vezet út (v levél a fában), akkor E(v)=Db(v), a v-ben termelt mennyiség, K(v)=1, M(v)=Db(v).

Ha v-be a v1,...,vt városokból megy út közvetlenül, akkor K(v)=K(v1)+...+K(vt); E(v)=Db(v)+E(v1)+...+E(vt); M(v)=Max{M(v1),...,M(vt)} ha Max{M(v1),...,M(vt)}*K(v)≤ E(v), E(v)/K(v) felső egészrésze egyébként.

A fenti összefüggések alapján E, K és M rekurzióval számítható. A feladat megoldása: `MinSzámol(1,E,K)`

```
MinSzámol(v,E,K):
  Ha BeFok(v)=0
     akkor E:=Db(v); K:=1; MinSzámol:=E
  különben E:=Db(v); K:=0; MaxM:=0
           Ciklus i=1-től BeFok(v)-ig
             vi:=Ut(v,i); Mvi:=MinSzámol(vi,Evi,Kvi)
             E:=E+Evi; K:=K+Kvi
             Ha Mvi>MaxM akkor MaxM:=Mvi
           Ciklus vége
           Ha K*MaxM≥E akkor MinSzámol:=MaxM
                     különben MinSzámol:=(E+K-1) Div K
  Elágazás vége
Eljárás vége.
```

Értékelési szempontok

A.	Egy út van	1 pont
B.	Minden út közvetlen 1-be megy	1 pont
C.	Minden út 1-ben találkozik csak	1 pont
D.	Két levél van és a találkozásban nagy érték	1 pont
E.	A levelekben nagy, a többi helyen 1 az érték	1 pont
F.	Mindenütt 1 az érték	1 pont
G.	Véletlen kis fa (N=20) azonos adatokkal	1 pont
H.	Véletlen kis fa (N=20) véletlen adatokkal	1 pont
I.	Véletlen közepes fa (N=50) azonos adatokkal	1 pont
J.	Véletlen közepes fa (N=50) véletlen adatokkal	2 pont
K.	Véletlen nagy fa (N=200) azonos adatokkal	2 pont
L.	Véletlen nagy fa (N=200) véletlen adatokkal	2 pont

5. feladat: Akadálypálya (15 pont)

A jármű minden időpontban valamely (x,y) koordinátájú pályaelemen áll, amelynek iránya i. Három lehetséges módon mozoghat tovább:

1. Átléphet egy szomszédos elemre, ha azon van sín és olyan irányban áll, ami csatlakozik.
2. A saját pályaelemét elfordítja.
3. Marad a helyén és elfordítja valamelyik szomszédos elemen a sínt, ha az nem áll csatlakozó irányban.

A harmadik esetben csak egy következő lépésben tud átlépni a szomszédos elemre. Így a jármű lehetséges helyzete a mozgás során megadható egy rendezett négyessel: (x,y,i,f), ahol

- (x,y): a pályaelem koordinátái, ahol a jármű van,
- i: a pályaelemen a sín iránya,
- f: hamis értéke azt jelenti, hogy a jármű valóban az (x,y) elemen áll, igaz értéke pedig fiktív állapotot jelez; azt, hogy egy szomszédos elemen áll és az (x,y) elemen a sínt már elfordította az i irányba.

Ha f értéke igaz (fiktív állapot), akkor a jármű csak egyet tehet, a következő lépésben, ténylegesen rálép az (x,y) elemre, ahol a sín i irányban áll.

Tekintve azt a gráfot, amelynek pontjai az (x,y,i,f) négyesek, élei pedig a leírt lépések szerint kapható elemek, a feladat egy legrövidebb út keresését jelenti a kiinduló elemről a jobb alsó sarokig. A megoldást szélességi bejárással végezzük.

```
Dx=((-1,+1),(0,0))  {a lépések vízszintesen}
Dy=((0,0),(-1,+1))  {a lépések függőlegesen}
Keres(Tól,Ig):
  Ciklus x=1-től M-ig
    Ciklus y=1-től N-ig
      NemVolt(x,y,igaz):=igaz; NemVolt(x,y,hamis):=igaz
    Ciklus vége
  Ciklus vége
  P.x:=Tól.x; P.y:=Tól.y; P.i:=Pálya(P.x,P.y)=1; P.f:=hamis
  P.idő:=0; NemVolt(P.x,P.y,P.i):=hamis; Ker:=-1
  SorInicializálás(S); Sorba(S,P)
  Ciklus amíg nem Üres?(S)
    Sorból(S,P); Q:=P
    Ha P.f
       akkor Ha NemVolt(Q.x,Q.y,Q.i)
                akkor Q.f:=hamis; Q.idő:=P.idő+1
                      Ha Ig.x=Q.x és Ig.y=Q.y
                         akkor Ker:=Q.idő
                      NemVolt(Q.x,Q.y,Q.i):=hamis; Sorba(S,Q)
             Elágazás vége
    különben Voltmár
  Ciklus vége
  Keres:=Ker
Függvény vége.
Voltmár:
  Ciklus l=1-től 2-ig          {lépés pályaelem irányába}
    Q.x:=P.x+Dx(Q.i,l); Q.y:=P.y+Dy(Q.i,l)
    Ha 0<Q.x és 0<Q.y és Q.x≤M és Q.y≤N és
                  Pálya(Q.x,Q.y)>0 és NemVolt(Q.x,Q.y,Q.i)
       akkor Q.idő:=P.idő+1
             Ha (Pálya(Q.x,Q.y)=1)=Q.i
                akkor Ha Ig.x=Q.x és Ig.y=Q.y
                         akkor Ker:=Q.idő
                      különben Q.f:=hamis
                               NemVolt(Q.x,Q.y,Q.i):=hamis
             különben Q.f:=igaz
             Sorba(S,Q)
    Elágazás vége
  Ciklus vége
  Q:=P; Q.i:=nem P.i     {az aktuális pályaelem elfordítása}
  Ha NemVolt(Q.x,Q.y,Q.i)
     akkor NemVolt(Q.x,Q.y,Q.i):=hamis
           Q.idő:=P.idő+1; Sorba(S,Q)
Eljárás vége.
```

Értékelési szempontok

A. Nincs üres pályaelem, és minden sín vízszintesen áll	1 pont
B. Nincs üres pályaelem, és minden sín függőlegesen áll	1 pont
3. Nem lehet eljutni	1 pont
4. Egyetlen út van	2 pont

5. Véletlen kis pálya (N=M=10) 2 pont

6. Véletlen közepes pálya (M=40,N=100) 2 pont

7. Véletlen nagy pálya (N=M=100) 2 pont

8. Van út, amelyen el lehet jutni úgy, hogy egy kanyarodáskor kell forgatni 2 pont

9. Van út, amelyen el lehet jutni úgy, hogy csak kanyarodáskor kell forgatni 2 pont

1999. Harmadik forduló

Ötödik-nyolcadik osztályosok

1. feladat: Kiszámolós (20 pont)

A kiesetteket kihúzzuk a tömbből, s úgy haladunk tovább:

```
Ciklus i=0-tól N-1-ig
  x(i):=i+1
Ciklus vége
j:=(k-1) mod N
Ciklus i=1-től N-1-ig
  Ki: x(j)
  Ciklus l=j-től N-2-ig
    x(l):=x(l+1)
  Ciklus vége
  n:=n-1; j:=(j+k-1) mod N
Ciklus vége
```

Értékelési szempontok

A. N=10, K=1-re jól írja ki a kiesőket 5 pont

B. N=10, K=3-ra az első körbeérésig jól írja ki a kiesőket 5 pont

C. N=10, K=3-ra végig jól írja ki a kiesőket és a végén megmaradtat 5 pont

D. N=5, K=6-ra végig jól írja ki a kiesőket és a végén megmaradtat 5 pont

2. feladat: Telefonáló robot (25 pont)

A kezdőpozíció a * jel fölött van. Ezután el kell mennünk számjegyenként a megfelelő helyre, majd a legvégén a végpozícióra, a # jel fölé.

```
x:=1; y:=4
Ciklus i=1-től hossz(szám)-ig
  Ha szám(i)='0' akkor xúj:=2; yúj:=4
                 különben Számmá(szám(i),j)
                          xúj:=1+(j-1) mod 3; yúj:=1+(j-1) div 3
  Ha xúj≠x akkor Ki: 'K ',xúj-x
  Ha yúj≠y akkor Ki: 'E ',y-yúj
  Ki: 'N '; x:=xúj; y:=yúj
Ciklus vége
xuj:=3; yuj:=4
Ha xuj≠x akkor Ki: 'K ',xúj-x
Ha yuj≠y akkor Ki: 'E ',y-yúj
```

Értékelési szempontok

1 → E 3 N K2 E -3 1+1 pont

9 → E 1 K 2 N E -1 1+1 pont

0 → K 1 N K 1 1+1 pont

1236547 → E 3 N K 1 N K 1 N	1+1+1 pont
E -1 N K -1 N K -1 N	1+1+1 pont
E -1 N E -1 K 2	1+2 pont
3571590 → E 3 K 2 N E -1 K -1 N E -1 K -1 N	2+1+1 pont
E 2 N E -1 K 1 N E -1 K 1 N	1+1+1 pont
E -1 K -1 N K 1	2+1 pont

3. feladat: Ismételgető (30 pont)

Vegyük az összes olyan K hosszúságú szót (K 2-től növekszik), amely legalább kétszer előfordul a szövegben! A végén rendezzük őket ábécésorrendbe!

```
db:=0
Ciklus k=2-től hossz(szó)-1-ig
  Ciklus i=1-től hossz(szó)-k-ig
    j:=i+1
    Ciklus amíg j≤hossz(szó)+1-k és
                szó(i..i+k-1)≠szó(j..j+k-1)
      j:=j+1
    Ciklus vége
    Ha j≤hossz(szó)+1-k
       akkor l:=1
             Ciklus amíg l≤db és
                         szó(x(l,1)..x(l,2))≠szó(i..i+k-1)
               l:=l+1
             Ciklus vége
             Ha l>db akkor db:=db+1; x(db,1):=i
                           x(db,2):=i+k-1
  Ciklus vége
Ciklus vége
Rendezés
```

Értékelési szempontok

A. Felismer 2 hosszú ismétléseket (mississippimississippi → ip,is,mi,pi,pp,si,ss)	4 pont
B. Mindegyiket csak egyszer írja ki (mississippimississippi → is,si,ss csak egyszer van)	3 pont
C. Az összes hosszabbat is felismeri (mississippimississippi → ip,ipp,ippi,is,iss,issi, és a leghosszabb> mississippi)	3 pont
D. A sorbarendezés ABC-szerint jó (mississippimississippi → ip,is,mi,pi,pp,si,ss)	4 pont
E. Az azonos kezdőszeletűekből a rövidebb van elöl (mississippimississippi → ip,ipp ,ippi,is,iss,issi,issip,issipp,issippi,issis,ississ,ississi,ississip,ississipp,ississippi,...)	4 pont
F. Nagyon hosszú is megy (qwertyuiopasdfghjklzxcvbnm 5-ször)	5 pont
G. Csupa azonos betűből álló is megy (aaa...a → aa,aaa,aaaa,...)	5 pont
H. Nincs egyáltalán ismétlődés (qwertyuiopasdfghjklzxcvbnm →)	2 pont

Kilencedik-tizedik osztályosok

1. feladat: Hálózati felügyelőprogram (23 pont)

Felhasználónként gyűjtsük ki az adatokat, s közben figyeljük az esetleges hibás műveleteket!

```
Időleképezés
  Ciklus i=-1-től végidő+1-ig
    idők(i):=0
  Ciklus vége
  Ciklus i=1-től adatdb-ig
    Felhasználókeresés(i,j,van)
    Ha t(i).irány='BE'
```

Belépők kezelése: Ismert felhasználó esetén hiba, ha nem lépett előzőleg ki, egyébként pedig felvesszük a belépési idejét. Új felhasználónál az egyéb adatait is kitöltjük.

```
        akkor Ha van
                 akkor Ha felh(j).beidő=-1
                          akkor felh(j).beidő:=t(i).idő
                       különben t(i).irány:='Be'
              különben felhdb:=felhdb+1
                       felh(felhdb).kód:=t(i).kód
                       felh(felhdb).idő:=0
                       felh(felhdb).beidő:=t(i).idő
    különben {ha t(i).irány='KI'}
```

Kilépők kezelése: Ismert felhasználó esetén hiba, ha nem lépett előzőleg be, egyébként pedig felvesszük az idejét. Új felhasználónál az egyéb adatait is kitöltjük és kezdőidőnek 0 óra 0 percet veszünk. Az adott percben gépen levők számát a teljes használati ideje alatt növeljük eggyel.

```
              Ha van
                 akkor Ha felh(j).beidő≠-1
                          akkor Ciklus k=felh(j).beidő-től
                                       t(i).idő-ig
                                  idők(k):=idők(k)+1
                                Ciklus vége
                                felh(j).idő:=felh(j).idő+
                                             t(i).idő
                                felh(j).beidő:=-1
                       különben t(i).irány:='Ki'
              különben Ciklus k=0-tól t(i).idő-ig
                         idők(k):=idők(k)+1
                       Ciklus vége

                       felhdb:=felhdb+1
                       felh(felhdb).kód:=t(i).kód
                       felh(felhdb).idő:=t(i).idő
                       felh(felhdb).beidő:=-1
  Ciklus vége
```

A nem kilépetteket a nap végéig bentlevőnek tekintjük:

```
  Ciklus i=1-től felhdb-ig
    Ha felh(i).beidő>-1
       akkor felh(i).idő:=felh(i).idő+végidő-felh(i).beidő
             Ciklus k=t(i).idő-től 24*60-1-ig
               idők(k):=idők(k)+1
             Ciklus vége
  Ciklus vége
Eljárás vége.
```

```
Felhasználókeresés:
  j:=1
  Ciklus amíg j≤felhdb és t(i).kód≠felh(j).kód
    j:=j+1
  Ciklus vége
  van:=(j≤felhdb)
Eljárás vége.
```

Az első részfeladatban a 0 szakaszok számát, kezdetét és végét kell megadni.

```
Foglaltidők:
  db:=0
  Ciklus i=0-tól végidő+1-ig
    Ha idők(i-1)=0 és idők(i)>0 akkor db:=db+1; kezd(db):=i
    különben ha idők(i-1)>0 és idők(i)=0 akkor vég(db):=i-1
  Ciklus vége
Eljárás vége.
```

A második részfeladat egy maximumkiválogatás:

```
Legjobbanfoglaltidők:
  max:=idők(0); md:=0
  Ciklus i=1-től végidő-ig
    Ha idők(i)>max akkor max:=idők(i)
  Ciklus vége
  Ciklus i=0-tól végidő-ig
    Ha idők(i-1)<max és idők(i)=max akkor md:=md+1; mk(md):=i
    különben ha idők(i-1)=max és idők(i)<max akkor mk(md):=i-1
  Ciklus vége
Eljárás vége.
```

A harmadik részfeladat egy egyszerű maximumkiválasztás:

```
Leghosszabbfelhasználó:
  max:=1; ldb:=0
  Ciklus i=1-től felhdb-ig
    Ha felh(max).idő<felh(i).idő akkor max:=i
  Ciklus vége
  Ciklus i=1-től felhdb-ig
    Ha felh(max).idő=felh(i).idő akkor ldb:=ldb+1; l(ldb):=i
  Ciklus vége
Eljárás vége.
```

A negyedik részfeladat a hibásak kigyűjtése. Az előfeldolgozásban átírtuk az irány kódjukat `BE`-ről `Be`-re, illetve `KI`-ről `Ki`-re, azaz már semmi egyéb teendőnk nincs.

Értékelési szempontok

A. Egy felhasználó kétszer jelentkezik be, hibás kijelentkezés	1+1+1+1 pont
B. Három felhasználó, átfedésekkel, a kétszer bejelentkező leghosszabb, hibás bejelentkezés	1+1+1+1 pont
C. Egyik felhasználó előző nap lép be, a másik pedig másnap lép ki	1+1+1+0 pont
D. Sok független intervallum, egyszeres belépések	1+1+1+0 pont
E. Sok, egyre hosszabb intervallum, azonos belépés	1+1+1+0 pont
F. Sok, mindkét végéről egyre szűkülő intervallum	1+1+1+0 pont
G. Sok átfedő intervallum, többszörös belépésekkel	1+1+1+0 pont

2. feladat: Katonák (28 pont)

Legyen O(x) a kezdetben az x-edik oszlopban, S(y) pedig az y-adik sorban álló katonák száma, KiHol(x,y) pedig az (x,y) helyen álló katona sorszáma vagy 0! Ebből számoljuk ki, hogy hányan állnak az i-edik oszlopig, illetve sorig!

```
Kezd:
  OIg(1):=O(1); SIg(1):=S(1)
  Ciklus i=2-től N-ig
    OIg(i):=OIg(i-1)+O(i); SIg(i):=SIg(i-1)+S(i)
  Ciklus vége
Eljárás vége.
```

A három részfeladatot egyszerre oldjuk meg. Amíg egy oszlopig pontosan annyian vannak, mint ahánynak addig lennie kell, addig nem csinálunk semmit:

```
E:=1; ps:=0
Ciklus amíg E≤N és Oig(E)=E
  E:=E+1
Ciklus vége
```

Külön vizsgáljuk azt az esetet, amikor egy oszlopig többen, illetve kevesebben vannak, mint ahányan kellenek. Egyik esetben jobbra, másik esetben pedig balra kell lépni katonáknak. Mindkét esetben meg kell keresni azt az oszlopot, ameddig pont annyi katona van, mint ahány kellene (ilyen biztosan van, mert pontosan N katona van)!

```
Ciklus amíg E<N
  U:=E
  Ha Oig(E)>E akkor Ciklus amíg Oig(U)>U
                      U:=U+1
                    Ciklus vége
           különben Ciklus amíg Oig(U)<U
                      U:=U+1
                    Ciklus vége
```

Kezdjük a balra léptetéssel:!

```
  Ha Oig(U)<U
     akkor Ciklus lx=E-től U-1-ig
```

Meg kell keresni E-től az első oszlopot, ahol van legalább 1 katona!

```
             x:=lx
             Ciklus amíg O(x)=0
               x:=x+1
             Ciklus vége
```

Meg kell nézni, hogy abban az oszlopban hol áll katona, és azt balra kell léptetni!

```
             y:=1
             Ciklus amíg KiHol(x,y)=0
               y:=y+1
             Ciklus vége
             KiHol(lx,y):=KiHol(x,y)
             KiHol(x,y):=0; O(x):=O(x)-1
             ps:=ps+1; P(ps):=(x,y,x-lx,'B')
           Ciklus vége
```

Amelyik oszlopok ettől kezdve rendben vannak, azokat átlépjük:

```
            E:=U
            Ciklus amíg E<N és Oig(E)=E
              E:=E+1
            Ciklus vége
```

Következik a jobbra léptetés:

```
  különben Ciklus lx=U-tól E+1-ig -1-esével
```

Meg kell keresni U-tól az utolsó oszlopot, ahol van legalább 1 katona!

```
              x:=lx
              Ciklus amíg O(x)=0
                x:=x-1
              Ciklus vége
```

Meg kell nézni, hogy abban az oszlopban hol áll katona, és azt jobbra kell léptetni!

```
              y:=1
              Ciklus amíg KiHol(x,y)=0
                y:=y+1
              Ciklus vége
              KiHol(lx,y):=KiHol(x,y)
              KiHol(x,y):=0; O(x):=O(x)-1
              ps:=ps+1; P(ps):=(x,y,x-lx,'J')
            Ciklus vége
```

Amelyik oszlopok ettől kezdve rendben vannak, azokat átlépjük:

```
            E:=U+1
            Ciklus amíg E<N és Oig(E)=E
              E:=E+1
            Ciklus vége
Ciklus vége
```

Most jönnek a sorok. Amíg egy sorig pontosan annyian vannak, mint ahánynak addig lennie kell, addig nem csinálunk semmit:

```
E:=0                          {igazodás soronként}
Ciklus amíg E≤N és Sig(E)=E
  E:=E+1
Ciklus vége
```

Itt felfelé, vagy pedig lefelé kell majd léptetni. Mindkettőhöz meg kell keresni azt a sort, ameddig pont annyi katona van, mint ahány kellene (ilyen biztosan van, mert pontosan N katona van)!

```
Ciklus amíg E<N
  U:=E
  Ha Sig(E)>E akkor Ciklus amíg Sig(U)>U
                      U:=U+1
                    Ciklus vége
            különben Ciklus amíg Sig(U)<U
                      U:=U+1
                    Ciklus vége
```

Kezdjük a lefelé léptetéssel!

```
  Ha Sig(E)<E
     akkor Ciklus ly=E-től U-1-ig
```

Meg kell keresni E-től az első sort, ahol van legalább 1 katona!

```
y:=ly
Ciklus amíg S(y)=0
  y:=y+1
Ciklus vége
```

Meg kell nézni, hogy abban a sorban hol áll katona, és azt lefelé kell léptetni!

```
x:=1
Ciklus amíg KiHol(x,y)=0
  x:=x+1
Ciklus vége
KiHol(x,ly):=KiHol(x,y); KiHol(x,y):=0
S(y):=S(y)-1
ps:=ps+1; P(ps):=(x,y,y-ly,'L')
Ciklus vége
```

Amelyik sorok ettől kezdve rendben vannak, azokat átlépjük:

```
E:=U
Ciklus amíg E<N és Sig(E)=E
  E:=E+1
Ciklus vége
```

Jön a felfelé léptetés:

```
különben Ciklus ly=U-tól E+1-ig -1-esével
```

Meg kell keresni U-tól az utolsó sort, ahol van legalább 1 katona!

```
y:=ly
Ciklus amíg S(y)=0
  y:=y-1
Ciklus vége
```

Meg kell nézni, hogy abban a sorban hol áll katona, és azt felfelé kell léptetni!

```
x:=1
Ciklus amíg KiHol(x,y)=0
  x:=x+1
Ciklus vége
KiHol(x,ly):=KiHol(x,y); KiHol(x,y):=0
S(y):=S(y)-1
ps:=ps+1; P(ps):=(x,y,y-ly,'F')
Ciklus vége
```

Amelyik sorok ettől kezdve rendben vannak, azokat átlépjük:

```
E:=U+1
Ciklus amíg E<N és Sig(E)=E
  E:=E+1
Ciklus vége
Ciklus vége
```

Már csak az A részfeladat megoldása hiányzik, azt kiszámolhatjuk az egyes katonák lépéseiből.

```
Lépsz:=0
Ciklus i=1-től ps-ig
  Lépsz:=Lépsz+P(i).h
Ciklus vége
```

Értékelési szempontok

A. Minden sorban 1 katona van, oszloponként kell tologatni	1+1+2 pont
B. Minden oszlopban 1 katona van, soronként kell tologatni	1+1+2 pont
C. Előbb soronként, majd oszloponként kell tologatni	1+1+2 pont
D. Előbb soronként, majd oszloponként kell tologatni, de nem akárkit	1+1+2 pont
E. A katonák a tábla közepén állnak	1+1+2 pont
F. A katonák a tábla négy sarkán állnak	1+1+2 pont
G. Nagy tábla, véletlenszerű elrendezés	1+1+2 pont

3. feladat: Fogadás (24 pont)

Legyen Ki(x) azok halmaza, akikkel x találkozott! Az az x lehet kezdőpont, amelyre $Ki(x)=\{p_1,...,p_k\}$ elemei elrendezhetőek úgy, hogy $Ki(p_1)\le...\le Ki(p_k)$. Az összefüggőség miatt pontosan 2 ilyen van.

```
x:=1
Ciklus amíg x≤N
  FSor(0):=Ki(x)∪{x}; fsz:=0; OK:=hamis; y:=1
  Ciklus amíg y≤N
    Ha y∈Ki(x)
       akkor k:=fsz; fsz:=fsz+1; H:=Ki(y)∪{y}
             Ha fsz=1
                akkor FSor(1):=H
             különben Ciklus amíg 0≤k és FSor(k)⊃H
                        FSor(k+1):=FSor(k); k:=k-1
                      Ciklus vége
                      Ha 0≤k és FSor(k)⊆H
                         akkor FSor(k+1):=H
                      különben y:=N+1
             Ok:=fsz=Fok(x)
    Elágazás vége
    y:=y+1
  Ciklus vége
  Ha Ok akkor P:=x; x:=N+1 különben x:=x+1
Ciklus vége
```

A további sorrend meghatározása: Legyen Volt a már besorolt elemek halmaza! Ha P az utolsónak besorolt elem, akkor a következő mindig a Ki(P)-Volt halmaz azon x eleme, amelyre Ki(x)-Volt elemszáma a legkisebb!

```
Sor(1):=P; Volt:=(P)
Ciklus i=2-től N-ig
  Min:=N+1
  Ciklus x=1-től N-ig
    Ha x∈Ki(P)-Volt
       akkor KixSzám:=0; H:=Ki(x)-Volt
             Ciklus y=1-től N-ig
               Ha y∈H akkor KixSzám:=KixSzám+1
             Ciklus vége
             Ha KixSzám<Min akkor Q:=x; Min:=KixSzám
    Elágazás vége
  Ciklus vége
  Sor(i):=Q; P:=Q; Volt:=Volt∪Q
Ciklus vége
```

Értékelési szempontok

A. Mindenki pontosan 2 emberrel találkozott, kivéve a két szélsőt	4 pont
B. Mindenki mindenkivel találkozott	4 pont
C. Mindenki K következővel találkozik	4 pont
D. Mindenki mindenkivel találkozó csoportok, akiket egy ember választ el	4 pont
E. Mindenki K következővel találkozik, nagy teszt	4 pont
F. Nagy véletlen teszt	4 pont

Tizenegyedik-tizenharmadik osztályosok

1. feladat: Elfogó (16 pont)

A feladat mélységi bejárással oldható meg. Tároljuk minden pontra, hogy a mélységi bejárással mikor jutottunk el hozzá (D vektor), valamint azt, hogy mely pontból jutottunk el hozzá (Apa vektor)! A rekurzió miatt néhány változót a főprogramban kell inicializálni.

```
Apa(1):=1; Sm:=0; Dd:=0; Bejár(1)

Bejár(P):
  Dd:=Dd+1; D(P):=Dd; Sm:=Sm+1; Sor(Sm):=P
  Ciklus i=1-től Fok(P)-ig
    Q:=G(P,i)
    Ha D(Q)=0 akkor Apa(Q):=P; Bejár(Q)
  Ciklus vége
  Ciklus i=1-től Fok(P)-ig     {a visszaélek feldolgozása}
    Q:=G(P,i)
    Ha Apa(Q)≠P és D(Q)>D(P)
       akkor Sm:=Sm+1; Sor(Sm):=Q; Sm:=Sm+1; Sor(Sm):=P
  Ciklus vége
  Ha P≠1 akkor Sm:=Sm+1; Sor(Sm):=Apa(P)  {a visszalépés P apjára}
Eljárás vége.
```

Értékelési szempontok

A. 1 lánc	2 pont
B. 1 fa	2 pont
C. 1 rács	2 pont
D. Nagy rács	2 pont
E. 3 rács, közös sarokponttal	2 pont
F. Sugaras, körgyűrűs hálózat	2 pont
G. Nagy véletlen gráf	4 pont

2. feladat: Ültetés (24 pont)

Tekintsük azt az irányítatlan gráfot, amelynek pontjai a székek, és p→q csak akkor v címkéjű él, ha a v vendég a p és a q helyeket igényelte! Az összefüggő komponenseket nézve azt láthatjuk, hogy ha a komponensben van kör, akkor minden szék felhasználható egy igény kielégítésére, és fordítva, a komponensbeli élek, mint igények közül legjobb esetben is pontosan annyi elégíthető ki, mint a komponens elemeinek száma.

Ha egy összefüggő komponensben nincs kör, azaz fa, akkor pontosan a fa pontjainak száma-1 igény elégíthető ki.

Mélységi keresést végezve, ha felfedezzük ez első kört (visszaélet), akkor azt megjegyezzük, és a bejárás után a kör egy pontját (elsőre megtalált) tesszük a komponens gyökerévé. Ha a feszítőfát nézzük, ahol a p→q éleket felfelé, a gyökér felé irányítjuk, akkor az igény hozzárendelés az lehet, hogy ha az él címkéje a v igény, akkor a v vendégnek a p széket adjuk.

Válasz a második kérdésre: Csak akkor lehet folytonosan foglalt helyekkel kielégíteni az igényeket, ha legfeljebb 2 összefüggő komponens lesz fa, és ekkor legfeljebb az az egy (vagy kettő) szék maradt ki a fenti elosztásban, amely(ek) a fá(k) gyökerei, és a legkisebb, legnagyobb helyek is a két fa valamelyikében vannak.

```
NemVolt:=(igaz,...,igaz); Faksz:=0
Ciklus P=1-től N-ig
  Ha Hely(P)=0
     akkor Bejár(P,k,v);     {egy összefüggő komponens feszítőfája}
           Ha VanKör akkor Fordít(P,k,v)
           különben Faksz:=Faksz+1
                    Ha Faksz=1
                       akkor gy1:=P; min1:=min; max1:=max
                    különben ha Faksz=2
                       akkor gy2:=P; min2:=min; max2:=max
Ciklus vége
```

Van-e folytonos helyfoglalásos megoldás?

```
Ha Faksz>2 akkor Összef:=hamis
különben Első:=0; Utolsó:=0; Üres:=0; U1:=0; U2:=0
         Ciklus P=1-től N-ig     {az üres helyek összeszámlálása}
           Ha Hely(P)>0
              akkor Ha Első=0 akkor Első:=P; Utolsó:=P
                                különben Utolsó:=P
           különben Ha Első>0 akkor Üres:=Üres+1
                                    Ha Üres=1
                                       akkor U1:=P
                                    különben Ha Ures=2
                                       akkor U2:=P
         Ciklus vége

         Összef:=hamis
         Elágazás Üres szerint
           0: Összef:=igaz
           1: Ha Faksz=1
                 akkor Összef:=igaz
                       ha gy1=U1 és min1=Első
                          akkor Fordít(gy1,min1,0)
                       különben ha gy1=U1 és max1=Utolsó
                                   akkor Fordít(gy1,max1,0)
           2: Ha Faksz=2
                 akkor Összef:=igaz
                       ha U1=gy1 és U2=gy2 vagy
                          U1=gy2 és U2=gy1
                       akkor ha Első=min1 és Utolsó=max2
                                akkor Fordít(gy1,min1,0)
                                      Fordít(gy2,max2,0)
                       különben Ha Első=min2 és Utolsó=max1
                                   akkor Fordít(gy1,max1,0)
                                         Fordít(gy2,min2,0)
        Elágazás vége
Elágazás vége
```

A bejárás a rekurzió miatt csak az inicializálásokat tartalmazza:

```
Bejár(P0,K,u):
  VanKör:=hamis; Hely(P0):=-1; min:=MaxN+1; max:=0
  MélyKeres(P0)
Eljárás vége.

MélyKeres(P):
  Ciklus i=1-től Fok(P)-ig
    v:=G(P,i)
    Ha NemVolt(v)
       akkor NemVolt(v):=hamis
             Ha P=Igény(v,1) akkor Q:=Igény(v,2)
                             különben Q:=Igény(v,1)
             Ha Q<min akkor min:=Q
             Ha max<Q akkor max:=Q
             Ha Hely(Q)=0 akkor Hely(Q):=v; MélyKeres(Q)
             különben Ha nem VanKör
                akkor VanKör:=igaz; u:=v; K:=Q
  Ciklus vége
Eljárás vége.

Fordít(P,k,v):   {a P←...←k úton megfordítjuk a Hely hozzárendelést}
  q:=k
  Ciklus amíg q≠P
    u:=Hely(q); Hely(q):=v
    Ha Igény(u,1)=q akkor Oq:=Igény(u,2)
                  különben Oq:=Igény(u,1)
    v:=u; q:=Oq
  Ciklus vége
  Hely(P):=v
Eljárás vége.
```

Értékelési szempontok

A. 1 folytonos lánc	1+1+1 pont
B. 1 nem folytonos lánc	1+1+1 pont
C. 1 kör	1+1+1 pont
D. 1 fa	1+1+1 pont
E. Több fa	1+1+1 pont
F. M független él	1+1+1 pont
G. K darab 2-szeres él; L darab több, mint kétszeres él	1+1+1 pont
H. Nagy véletlen teszt	1+1+1 pont

3. feladat: Autómentés (35 pont)

Első lépésként az autópályát (`AP0` tömb) üresre állítjuk, az első belépőket beléptetjük (az `AP0` tömbben megjelenik a megfelelő helyen a sebességük), majd szimuláljuk a forgalmat az első érkezőtől a baleset idejéig (`Bt`), s ebből megkapjuk a baleset helyét (`Bx`). A belépő autók adatai az `A` tömbben vannak.

```
Kezd:
  AP0:=((0,...,0),...,(0,...,0)); Asor:=1; t0:=1
  Ciklus amíg A(Asor).t>t0
    t0:=t0+1
  Ciklus vége
  Ciklus amíg A(Asor).t=t0
    AP0(1,A(Asor).s):=A(Asor).v; Asor:=Asor+1
  Ciklus vége
  t:=t0; OK:=hamis
  Ciklus amíg t<Bt
    t:=t+1; Szimulál(hamis,t,AP0,1,Bx)
  Ciklus vége
Eljárás vége.

Szimulál(C,t,AP,Tol,Ig):
  Uj:=igaz
  Ciklus s=1-től Ms-ig
    R(hamis,s):=0; v:=AP(Ig,s)
    Ha v>0 akkor E(s):=Ig; x:=Ig+AP(Ig,s); K(s):=x+1
    különben K(s):=Bx+2; x:=Ig-1        {nincs autó}
             Ciklus amíg x>0 és AP(x,s)≤0
               x:=x-1
             Ciklus vége
             E(s):=x
  Ciklus vége
  x:=Ig
  Ciklus amíg x≥Tol
    Lép(x); Ha OK akkor x:=0 különben x:=x-1
  Ciklus vége
                  {a t időben érkező A(t) autók beléptetése a pályára AP(1)-be}
  Ciklus amíg Asor≤Asz és A(Asor).t=t
    AP(1,A(Asor).s):=A(Asor).v; Asor:=Asor+1
  Ciklus vége
Eljárás vége.

Lép(x):            {Minden sáv x pozíciójában levő autót lépteti}
  R(Uj):=AP(x); UjK:=(-1,...,-1)
  Ciklus s=1-től Ms-ig
    v:=AP(x,s)
    Ha v>0 akkor Autó          {autó van (x,s)-ben}
    különben ha v<0 akkor Mentő   {v<0; mentő van az x,s pozícióban}
  Ciklus vége
  Ciklus s=1-től Ms-ig         {E és K aktualizálása}
    v :=R(Uj,s)
    Ha v>0 akkor x1:=x-1           {autó van (x,s)-ben}
                 Ciklus amíg x1>0 és AP(x1,s)≤0
                   x1:=x1-1
                 Ciklus cége
                 E(s):=x1
    Elágazás vége
    Ha UjK(s)≥0 akkor K(s):=UjK(s)
    AP(x,s):=0
  Ciklus vége
  Uj:=nem Uj
Eljárás vége.
```

```
Autó:
  Ha R(nem Uj,s)≤0 vagy R(nem Uj,s)>0 és v≤R(nem Uj,s) vagy C
     akkor x1:=Min(x+v,K(s)-1); AP(x1)(s):=v {v előtt nincs autó; előre}
           Ciklus xx=x+1-től x1-1-ig
             AP(xx,s):=0;          {mentők törlése}
           Ciklus vége
           UjK(s):=x1
  különben Ha s<Ms és E(s+1)+AP(E(s+1),s+1)≤x {sávváltás}
        akkor AP(x+1,s+1):=v; UjK(s+1):=x+1 {balra?}
  különben ha s>1 és AP(x+1,s-1)≤0 és E(s-1)+AP(E(s-1),s-1)≤x
                 akkor AP(x+1,s-1):=v; UjK(s-1):=x+1 {jobbra?}
  különben x1:=Min(x+v,K(s)-1); AP(x1)(s):=v    {Nem tud előzni}
           Ciklus xx=x+1-től x1-1-ig
             AP(xx,s):=0             {mentők törlése}
           Ciklus vége
           UjK(s):=x1
Eljárás vége.

Mentő:
  x0:=E(s)+AP(E(s),s)              {Előre}
  Ha x0>x akkor x0:=x0+1 különben x0:=x+1
  x1:=Min(x+Mv,K(s)-1); xx:=x0
  Ciklus amíg xx≤x1
    AP(xx,s):=-1
    Ha s=1 és x<Bx és xx≥Bx akkor OK:=igaz; xx:=+∞
                           különben xx:=xx+1
  Ciklus vége

  Ha s>1 és AP(x+1,s-1)≤0 és E(s-1)+AP(E(s-1),s-1)≤x
     akkor AP(x+1,s-1):=-1     {sávváltás jobbra}
  Ha s<Ms és AP(x+1,s+1)≤0 és E(s+1)+AP(E(s+1),s+1)≤x
     akkor AP(x+1,s+1):=-1     {sávváltás balra}
Eljárás vége.
```

Az A részfeladat megoldásához meg kell számolni és ki kell gyűjteni a `Bt` időpontban a `Bx` pozícióig levő autókat.

```
A_Feladat:
  Szám:=0
  Ciklus s=1-től Ms-ig
    Ciklus x=1-től Bx-ig
      Ha AP0(x,s)>0 akkor Szám:=Szám+1; Hol(Szám).x:=x
                          Hol(Szám).y:=s
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

A B részfeladatban az `Ap` tömbben megadjuk, hogy mely pozíción állnak éppen az autók (ezek nem mozdulhatnak), s a mentőket kell csak szimulálnunk:

```
B_Feladat:
  Ciklus x=1-től Bx-ig
    Ciklus s=1-től Ms-ig
      Ha AP0(x,s)>0 akkor AP(x,s):=-1      {itt autó van}
               különben AP(x,s):=+∞;  {ide még nem juthatott el mentő}
      Ut(x,s):=0
    Ciklus vége
  Ciklus vége
```

Most elhelyezünk minden sáv 1. pozíciójára egy-egy mentőt, s megnézzük, hogy meddig haladhat abban a sávban (KovA tömb):

```
Ciklus s=1-től Ms-ig   {sávonként a következő autó meghatározása}
  AP(1,s):=1;            {mentők az 1. pozícióban a Bt+1 időben}
  AP(Bx+1,s):=+∞; x:=1
  Ciklus amíg x<Bx+1 és AP(x,s)≥0
    x:=x+1
  Ciklus vége
  KovA(s):=x           {eddig mehet előre}
Ciklus vége
```

Ezután haladunk pozíciónként, azon belül sávonként, amíg egy mentőautó a baleset helyszínére nem ér:

```
OK:=hamis; x:=1
Ciklus amíg x≤Bx és nem OK
  s:=1
  Ciklus amíg s≤Ms és nem OK
    Ha AP(x,s)=-1            {autó áll a sávban}
       akkor x1:=x+1
             Ciklus amíg x1<Bx+1 és AP(x1,s)≥0
               x1:=x1+1
             Ciklus vége
             KovA(s):=x1     {eddig mehet előre}
    különben ha AP(x,s)<+∞        {mentő: lépés előre}
       akkor x1:=Min(x+Mv,KovA(s)-1)
             Ciklus xx:=x+1-től x1-ig
               Ha AP(x,s)+1<AP(xx,s)
                  akkor AP(xx,s):=AP(x,s)+1
                        Ut(xx,s):=xx-x
                        Ha s=1 és xx≥Bx {odaér}
                           akkor OK:=igaz; vegx:=xx
             Ciklus vége

             Ha s<Ms és AP(x+1,s+1)≥0 és    {sávváltás balra}
                           AP(x,s)+1<AP(x+1,s+1)
                akkor AP(x+1,s+1):=AP(x,s)+1
                      Ut(x+1,s+1):=-1
             Ha s>1 és AP(x+1,s-1)≥0) és     {sávváltás jobbra}
                           AP(x,s)+1<AP(x+1,s-1)
                akkor AP(x+1,s-1):=AP(x,s)+1
                      Ut(x+1,s-1):=-2
    Elágazás vége
    s:=s+1
  Ciklus vége
  x:=x+1
Ciklus vége
Ha OK akkor Ki: AP(vegx,1); UtKiIr(Ut,Vegx)
   különben Ki: -1
Eljárás vége.
```

A C és a D feladat nagyon hasonlít egymásra, ezért közös megoldást készítünk, amit kétszer kell meghívni: a C részfeladatnál a C paraméter értéke legyen igaz, a D részfeladatnál pedig hamis értékű! A baleset időpontjában be kell állítani az autók sebességét, majd a baleset után szimulálunk, amíg egy mentő a balesethez nem ér:

```
CD_Feladat(C):
  Ciklus s=1-től Ms-ig
    Ciklus x=0-tól Bx+1-ig
      Ha AP0(x,s)>0            {autó sebesség beállítás}
         akkor Ha C akkor AP(x,s):=Mv-1
                  különben AP(x,s):=AP0(x,s)
      különben AP(x,s):=0
    Ciklus vége
  Ciklus vége
  t:=Bt+1; OK:=hamis; Szimulál(C,t,AP,1,Bx)
  Ciklus s=1-től Ms-ig
    AP(1,s):=-1;       {a mentők lehetséges pozíciói Bt+1 időben }
  Ciklus vége
  Ciklus amíg nem OK
    t:=t+1; Szimulál(C,t,AP,1,Bx)
  Ciklus vége
  Ha OK akkor Ki: t-Bt különben Ki: -1
Eljárás vége.
```

Értékelési szempontok

A. Van szabad sáv, azonos sebességű autók	1+1+1+1+1+1 pont
B. Különböző sebességű autók, nincs sávváltás, leállva elzárják az utat	1+1+1+0+0+0 pont
C. Van sávváltás is, leállás esetén elzárják a baleset helyszínét	1+1+1+0+0+0 pont
D. Véletlen teszt	1+1+1+1+1+2 pont
E. Nagy véletlen teszt	1+1+1+1+2+2 pont
F. Sűrű szélső sáv, ritkább belsők	1+1+1+1+2+2 pont