10. feladat: Csővezeték (28 pont)

Egy országban ***K*** helyen építettek olajfinomítót, ahol kőolajból benzint készítenek. ***N*** helyen van benzinkút, ahova a benzint el kell juttatni csővezetéken valamelyik olajfinomítóból. Úgy kell megtervezni a csővezetéket, hogy minimális hosszúságú csövet kelljen lefektetni. Csővezeték csak benzinkútnál vagy finomítónál ágazhat el, a térképen csak vízszintesen vagy függőlegesen vezethető, a vezetékek keresztezhetik egymást, de csatlakozás csak finomítónál vagy benzinkútnál lehet. A finomítókat 1-től K-ig, a benzinkutakat pedig K+1-től K+N-ig sorszámozzuk.

Feladat

Készíts programot (CSO.PAS vagy CSO.C), amely meghatározza, hogy mely csomópontok (olajfinomító vagy benzinkút) között kell kiépíteni a csővezetéket úgy, hogy a csővezeték hossza minimális legyen!

Bemenet

A CSO.BE állomány első sorában az olajfinomítók ($1 \le K \le 10$) és a benzinkutak ($1 \le N \le 1000$) száma van egy szóközzel elválasztva. A következő K sor az egyes olajfinomítók, az ezt követő N sor pedig a benzinkutak koordinátáit tartalmazza. Minden sorban egy sor- és egy oszlopindex szerepel (-1000 és 1000 közötti egész számok), egymástól egy szóközzel elválasztva.

Kimenet

A CSO.KI állomány első sorába a minimális hosszúságú csőhálózat hosszát kell írni, a további sorokba pedig azon helyek indexeit, amelyek között csövet kell fektetni a minimális hosszúságú csőhálózat kiépítéséhez. Minden sorban egy összekötés két végének sorszáma szerepeljen!

Példa:

CSO.BE	CSO.KI
2 4	400
100 100	2 3
0 100	1 4
-100 100	2 5
100 0	4 6
40 40	
150 -50	

11. feladat: Robotváros (42 pont)

Robot város úthálózata olyan, hogy minden kereszteződésben pontosan 3 út találkozik. Azt mondjuk, hogy az A—B úttól C jobbra, D pedig balra van, ha az óramutató járásával szemben haladva az ABC szög kisebb, mint az ABD szög. Egy robotot kell irányítanunk a labirintusban a J és a B betűk, mint parancsok segítségével, melyek azt jelentik, hogy az adott kereszteződésben a robotnak jobbra vagy balra kell fordulnia.

Feladat:

Készíts programot (VAROS.PAS vagy VAROS.C), amely egy parancssorozattal megadja azt az utat, amely a legkevesebb/legtöbb kereszteződést érintve vezet az A kereszteződésből a V kereszteződésbe úgy, hogy a robot minden kereszteződésen és minden úton legfeljebb egyszer halad át!

Bemenet:

A VAROS.BE állomány első sorában a kereszteződések száma (***1≤N≤100***) van. A második sorban a robot tartózkodási helye (***1≤A≤N***) és annak a kereszteződésnek a sorszáma van, ahova először lép (***1≤B≤N***). A harmadik sor a robot által elérendő kereszteződés sorszámát (***1≤V≤N***) tartalmazza. A következő ***N*** sor az egyes kereszteződések X és Y koordinátáját, valamint a vele szomszédos három kereszteződés sorszámát tartalmazza, egy-egy szóközzel elválasztva.

Kimenet:

A VAROS.KI állományba két sort kell írni. Az elsőbe azt a parancssorozatot, amely a robotot az ***A***-ból a ***B*** felé indulva a ***V*** kereszteződésbe viszi a lehető legkisebb, a másodikba pedig azt, ami a lehető legnagyobb lépésszámú olyan úton, ami a fenti szabályoknak megfelel.

Példa:

VAROS.BE

```
8
1 2
5
0 0 2 3 7
10 0 1 4 8
3 3 1 4 5
7 3 2 3 6
3 7 3 6 7
7 7 4 5 8
0 10 1 5 8
10 10 2 6 7
```

VAROS.KI

```
JJB
BJJBB
```

12. feladat: Strucctojás (30 pont)

Van néhány teljesen egyforma strucctojásunk, és egy sokemeletes panelház. El szeretnénk dönteni, hogy a ház hányadik emeletéről lehet még kidobni egy tojást úgy, hogy ne törjön össze. A tojások „jól viselkednek”, azaz ha valahányadikról egyszer kidobtunk egyet anélkül, hogy összetört volna, akkor minden annál nem nagyobb sorszámú emeletről kidobva egy tojást, az épen marad. Egy tojás, ha nem tört össze egy kísérlet során, akkor mindenfajta károsodás nélkül élte túl azt, azaz akár a legmagasabban fekvő – a tojások számára még elviselhető – emeletről kidobva is épen marad. (Azaz nem „gyűjti” a részleges sérüléseket.) Ezért nem is tartjuk számon, hogy mikor melyik tojással próbálkozunk – nem is tudnánk őket megkülönböztetni, hiszen olyan egyformák, mint két tojás –, csak azt, hogy hány még épen maradt tojásunk van. Korlátozott számú tojásunk van; ha az mind összetört, akkor már választ kell adnunk. A cél az, hogy minél kevesebb próbálkozással választ adjunk.

A rendelkezésre álló tojások száma $1 \le K \le 8$, a ház emeleteinek száma $1 \le N \le 1'900'000'000$ (legfeljebb 1,9 milliárd). Minden teszteset olyan, hogy bármely megoldás kitalálható legfeljebb 700 dobással.

A kísérleteket egy tárgykódban rendelkezésre álló modul segítségével végezhetik a programok. Ennek neve Pascal esetén `strhaz.tpu`, C változata `strhaz.obj`, illetve `strhaz.h`. Ezeket az állományokat a floppylemezek gyökérkönyvtárában helyeztük el.

A program (`STRUCC.PAS` vagy `STRUCC.C`) az N, K bemeneti adatokat is a modultól kapja, **semmilyen fájlt nem olvashat és nem is írhat.**

A következő függvények/eljárásokat lehet használni:

```
Function EmeletekSzama:LongInt;
extern long EmeletekSzama();

Function TojasokSzama:byte;
extern unsigned char TojasokSzama();

Function Dob(emelet:LongInt):boolean;
extern int Dob (long emelet);
```

Az egyetlen paraméterrel az emelet sorszámát kell megadni, ahonnan kidobjuk a tojást. Az emeleteket 1-től `N`-ig sorszámozzuk.

Pascalban `true`, C-ben `1` visszaadott érték esetén a tojás összetört; Pascalban `false`, C-ben `0` visszatérési érték esetén a tojás megmaradt.

```
Procedure Valasz(emelet:LongInt);
extern void Valasz (long emelet);
```

Ennek az eljárásnak a meghívásával kell a megoldást közölni. Az egyetlen paraméter annak az emeletnek a sorszámát tartalmazza, amelyről még ki lehet dobni a tojást anélkül, hogy összetörne. Ha már az elsőről kidobva is összetörik a tojás, akkor a válasz 0 legyen. Az eljárás nem tér vissza, hanem a kimeneti fájl megírása után megszakítja a program futását.

Modul használata Pascalban

A lemezen adott `strhaz.tpu`-t az aktuális könyvtárba kell másolni, és a főprogram elejére a `uses strhaz;` parancsot kell beírni a következőképpen:

```
Program Strucc;
Uses StrHaz;
```

Majd használhatjuk a feljebb deklarált eljárásokat úgy, mintha magunk írtuk volna meg őket.

Modul használata C-ben:

A lemezen található `strhaz.obj` és `strhaz.h` nevű fájlokat az aktuális könyvtárba kell másolni. A főprogram elejére be kell írni a következő sort:

```
#include "strhaz.h"
```

Majd a Project|Open project menüponttal meg kell nyitni egy pojektet és a Project|Add file menüponttal hozzá kell adni a `strucc.c` és a `strucc.obj` fájlokat. A `strucc.c` programszöveg ablakából nem lehet közvetlenül kiadni a Build vagy Make parancsot, mert linkelési hibával leáll. Ehelyett, ha `.EXE` állománnyá akarjuk fordítani a programot, akkor ezt a parancsot a Project ablakban kell kiadni.

Megjegyzés: Erősen ajánljuk a C nyelv használatát a C++ helyett. Aki mégis az utóbbival szeretne dolgozni, használja a floppylemez `\STRHCPP` alkönyvtárában található `strhaz.obj` fájlt.

Tesztelés

A verseny ideje alatt `STRUCC.BE` bemeneti fájlok készítésével és az aktuális könyvtárba helyezésével lehet tesztelni a programot. Ez a fájl egyetlen sorában három T, N, K egész számot tartalmaz egyetlen szóközzel elválasztva: a rendelkezésre álló tojások számát, a ház magasságát és a megoldást, azaz a legmagasabban fekvő olyan emelet sorszámát, melyről még ki lehet dobni a tojást anélkül, hogy összetörne.

Helyes futás esetén a modul készít egy `STRUCC.KI` nevű kimeneti fájlt, mely három számot tartalmaz szóközzel elválasztva: az első szám mindig 0, a második a meghívott `Valasz` eljárásnak megadott paraméter, a harmadik pedig az elvégzett kísérletek száma.