Kedves Versenyző! A megoldások értékelésénél csak **a programok futási eredményeit** vesszük tekintetbe. Ezért igen fontos a **specifikáció pontos betartása**. Ha például a feladat szövege adatok valamilyen állományból történő beolvasását írja elő, és a program ezt nem teljesíti, akkor a feladatra nem adunk; az objektív értékelés érdekében a pontozóknak a programszövegekben egyetlen karaktert sem szabad javítaniuk, az előre megadott javítási útmutatótól semmiben nem térhetnek el. A programokat csak a feladatkiírásban leírt szabályoknak megfelelő adatokkal próbáljuk ki, emiatt nem kell ellenőrizni a bemenő adatok helyességét, illetve a szükséges állományok létezését. Ha a programnak valamilyen állományra van szüksége, akkor azt mindig az aktuális könyvtárba kell rakni. Az állományok neve minden esetben rögzített. **Csak olyan programokat értékelünk, amelyek 1 percen belül adnak végeredményt!**

1. feladat: Fal (44 pont)

A Kínai Nagy Falon N őrhelyet létesítettek. Közülük azonban csak M helyen van őrség. Két szomszédos őrhely közötti fal őrzött, ha legalább az egyik végén van őrség; védett, ha mindkét végén van őrség. Őrzött szakasznak nevezzük egymást követő őrzött falak nem bővíthető sorozatát. Hasonlóan, védett szakasznak nevezzük egymást követő védett falak nem bővíthető sorozatát.

Készíts programot (`fal.pas`, …), amely megadja a védett és az őrzött szakaszok számát, valamint azt, hogy minimum hány helyre kell még őrséget küldeni, hogy minden fal őrzött legyen!

A `fal.be` szöveges állomány első sorában az őrhelyek száma ($1 \le N \le 100$) és az őrségek száma ($1 \le M \le 100$) van, egy szóközzel elválasztva. A következő M sor az őrségek leírását tartalmazza, közülük az i-edik annak az őrhelynek a sorszáma, ahol az i-edik őrség van. Tudjuk, hogy minden helyen legfeljebb 1 őrség van.

A `fal.ki` szöveges állományba három sort kell írni! Az első sorba a védett szakaszok száma, a másodikba az őrzött szakaszok száma kerüljön! A harmadik sorba az új őrségek minimális számát kell írni, amivel elérhető, hogy minden fal őrzött legyen!

Példa:

`fal.be`

```
15 9
6
3
12
11
4
5
8
15
14
```

`fal.ki`

```
3 [3-6. őrhely, 11-12. őrhely, 14-15. őrhely]
2 [2-9. őrhely, 10-15. őrhely]
2 [az 1. és a 9. őrhelyre]
```

2. feladat: Utcák (30 pont)

Egy modern nagyváros úthálózata egy négyzetráccsal írható le, ahol N jelöli a négyzetrács sorainak számát (azaz a kelet-nyugati irányú utak számát, az ilyen utakat alulról felfelé sorszámozzuk), M pedig az oszlopokét (azaz az észak-déli utakét, az ilyen utakat balról jobbra sorszámozzuk). El szeretnénk jutni a város egyik kereszteződéséből egy másik kereszteződésbe. Az egyes kereszteződésekből kivezető néhány út elejére behajtani tilos táblákat helyeztünk el, arra értelemszerűen nem lehet haladni.

Útközben a várost nem hagyhatjuk el (bár erről szóló jelzőtáblák nincsenek).

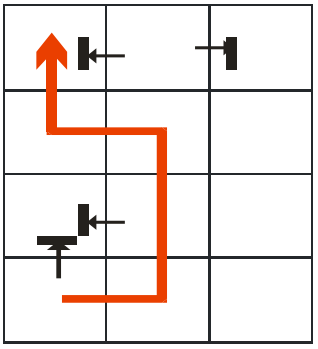
Írj programot (`utcak.pas`, …), amely a táblák figyelembevételével megadja a legrövidebb útvonalat, amelyeken áthaladva eljuthatunk az indulási helyről a célba!

A `utcak.be` állomány első sorában a sorok és oszlopok száma ($1 \le N,M \le 100$), valamint a táblák száma ($1 \le T \le 10\,000$) van egy-egy szóközzel elválasztva. A következő T sorban soronként egy-egy tábla leírása található. A táblaleírás formája: sor_1 $oszlop_1$ sor_2 osz-

lop_2, ahol a sor_i és az $oszlop_i$ két szomszédos csomópont koordinátáit adja meg ($1 \le sor_i \le N$, $1 \le oszlop_i \le M$). Jelentése: a (sor_1,$oszlop_1$)-ből a (sor_2,$oszlop_2$)-be vezető útra behajtani tilos tábla van. Az utolsó sorban a két kereszteződés sor és oszlopindexe van, egy-egy szóközzel elválasztva, az első az induló hely, a második a cél.

A `utcak.ki` állomány első sorába a két pont közötti legrövidebb út hosszát kell írni! A második sorba egy legrövidebb út leírását kell írni, ahol minden lépést a haladás iránya, azaz az E, K, D vagy N betű azonosít. Feltehető, hogy ilyen út mindig van!

Példa:

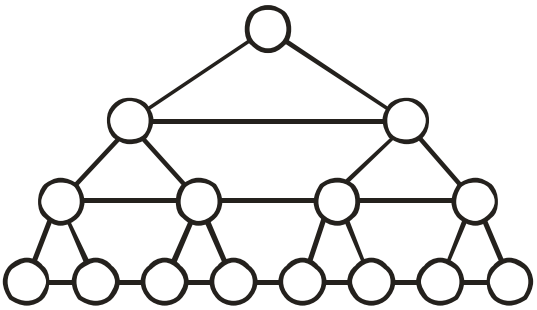


```
utcak.be      utcak.ki
4 3 4         5
1 1 2 1       KEENE
2 2 2 1
4 2 4 1
4 2 4 3
1 1 4 1
```

`1 1 2 1` magyarázata: Az (1,1) pontból a (2,1) pont felé vezető úton behajtani tilos tábla van, arra nem mehetünk.

3. feladat: Játéktábla (40 pont)

Egy játéktábla 101 sorból áll, minden sorában pontosan kétszer annyi elem van, mint a fölötte levő sorban. A tábla a következő szerkezetű:



A tábla felső pontjából indulunk. Az egyes lépéseket a következők írják le:

- 0 balra lefelé lépünk egyet,
- 1 jobbra lefelé lépünk egyet,
- 2 felfelé lépünk egyet,
- 3 balra lépünk egyet,
- 4 jobbra lépünk egyet.

Készíts programot (`tabla.pas`, …), amely beolvas egy lépéssorozatot, amely elvezet a tábla valamely eleméhez, majd megad egy olyan lépéssorozatot, amely a legrövidebb úton vezet ugyanide!

A `tabla.be` állomány első sorában a lépések K száma van ($1 \le K \le 100$), a következő sorban pedig az egyes lépéseket leíró K darab szám, egy-egy szóközzel elválasztva. A lépéssorozat biztosan helyes, azaz nem hagyjuk el vele a játéktáblát.

A `tabla.be` állomány első sorába a legrövidebb lépéssorozat L hosszát kell írni, amely a bemenetben kapott lépéssorozattal azonos helyre vezet! A második sorba pedig egy ilyen legrövidebb lépéssorozat kerüljön, azaz L szám, egy-egy szóközzel elválasztva!

Példa:

```
tabla.be          tabla.ki
6                 3
0 1 4 2 1 0       1 1 0
```

4. feladat: Mozi (36 pont)

Egy nagyon várt film vetítésére a szervező jegyrendeléseket fogad. Minden igénylő egy jegyet igényelhet, az igénylésben megad egy ülőhely sorszámot. A feltétel az, hogy ha egy igénylő az igényében az ***s*** sorszámot adta meg, akkor el kell fogadnia olyan ***u*** sorszámú ülőhelyet, amelyre teljesül, hogy $s \le u \le s+K$, ahol ***K*** egy előre rögzített nemnegatív szám. A szervező feladata, hogy az igénylések közül kiválassza azt a legtöbb igényt, amelyet ki tud elégíteni. Bármely ülőhelyet legfeljebb egy igénylő kaphat meg.

Készíts programot (`mozi.pas`, …), amely kiszámítja, hogy legjobb esetben hány igénylő kérését lehet kielégíteni! A program adjon is meg egy megfelelő jegykiosztást!

A `mozi.be` szöveges állomány első sorában három egész szám van, az ülőhelyek ***M*** száma ($1 \le M \le 3000$), az igények ***N*** száma ($1 \le N \le 10000$) és a ***K*** ($0 \le K \le 100$) értéke. Az ülőhelyeket az ***1,…,M*** számokkal azonosítjuk. A második sor pontosan ***N*** egész számot tartalmaz (egy-egy szóközzel elválasztva). Az i-edik szám annak az ülőhelynek a sorszáma, amelyet az i-edik igénylő szeretne megkapni.

A `mozi.ki` szöveges állomány első sora egy ***L*** egész számot tartalmazzon, a legtöbb kielégíthető igény számát! A következő ***L*** sor egy megfelelő jegykiosztást tartalmazzon. Minden sorban két egész szám legyen egy szóközzel elválasztva, az első szám egy igénylő sorszáma, a második pedig annak az ülőhelynek a sorszáma legyen, amelyiket ez az igénylő kap! A kiírás sorrendje tetszőleges. Több megoldás esetén bármelyik megadható.

Példa:

`mozi.be`

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

`mozi.ki`

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

Elérhető összpontszám: 150 pont + 50 pont az 1. fordulóból