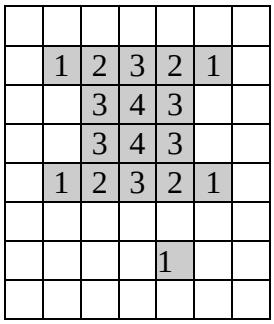
**Kedves Versenyző!** A megoldások értékelésénél csak **a programok futási eredményeit** vesszük tekintetbe. Ezért igen fontos a **specifikáció pontos betartása**. Ha a feladat szövege adatok valamilyen állományból történő beolvasását írja elő, és a program ezt nem teljesíti, akkor a feladatra nem adunk pontot (akkor sem, ha egyébként tökéletes lenne a megoldás); az objektív értékelés érdekében ugyanis a programszövegekben egyetlen karaktert sem javítunk, s az előre megadott javítási útmutatótól semmiben nem térünk el. A programokat csak a feladatkiírásban leírt szabályoknak megfelelő adatokkal próbáljuk ki, emiatt nem kell ellenőrizni, hogy a bemenő adatok helyesek-e, illetve a szükséges állományok léteznek-e (sőt ezért plusz pont sem jár). Ha a programnak valamilyen állományra van szüksége, akkor azt mindig az aktuális könyvtárba kell rakni. Az állomány neve minden esetben rögzített.

1. feladat: Jégtömb (25 pont)

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




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

Készíts programot **`jegtomb.cbp (main.cpp)`**, amely egy adott jégtömbre megadja, hogy hány időegység alatt olvad el teljesen, illetve időegységenként megadja, hogy a jégtömb még hány mezőből áll!

Bemenet

A **`jegtomb.be`** állomány első sorában két egész szám van, egyetlen szóközzel elválasztva, a táblázat sorainak ($1 \le N \le 100$) és oszlopainak ($1 \le M \le 100$) a száma. A következő N sor mindegyike M számjegyet tartalmaz, közülük az i-edik sor j-edik számjegye 0, ha a táblázat i-edik sorának j-edik oszlopában levegő van, és 1, ha jég. A táblázat szélső soraiban és oszlopaiban biztosan levegő van.

Kimenet

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

Példa: (az ábrán látható jégtömbre)

| jegtomb.be | jegtomb.ki |
|---|---|
| 8 7 | 4 |
| 0000000 | 17 |
| 0111110 | 12 |
| 0011100 | 8 |
| 0011100 | 2 |
| 0111110 | |
| 0000000 | |
| 0000100 | |
| 0000000 | |

2. feladat: Licit (25 pont)

Nevesincs város polgármestere elhatározta, hogy értékesíti a város mellett levő téglalap ala-

kú földdarabot. A földet egyforma méretű parcellákra osztotta.

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

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

Írj programot **licit.cbp (main.cpp)**, amely megadja a választ a polgármester problémájára!

Bemenet:

A **licit.be** állomány első sorában a pályázatok N száma (1≤N≤100) és a parcellák M száma (1≤M≤100) található, egy szóközzel elválasztva. A következő N sor az egyes pályázók adatait tartalmazza, a pályázókat ez a sorrend azonosítja. Mindegyik sorban 3 szám van egy-egy szóközzel elválasztva: A B FT, ami azt jelenti, hogy a pályázó az A sorszámú parcellától (1≤A≤M) a B sorszámú parcelláig (A≤B≤M) terjedő részért FT forintot fizetne (1000≤FT≤ 1000000).

Kimenet:

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

Példa:

```
licit.be              licit.ki
4 5                   11000
1 5 10000             2 4
2 3 5000
4 5 5000
4 4 6000
```

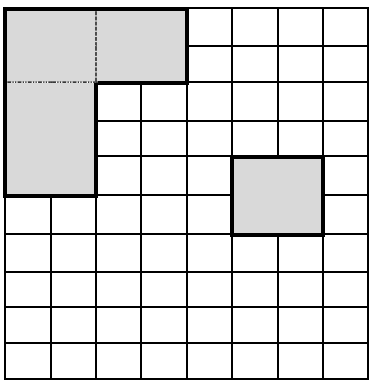
3. feladat: Térkép (25 pont)

Újlandia térképének (N sorból, M oszlopból álló téglalap) K db téglalap alakú részéről készítettünk nagyításokat. A téglalapok oldalai párhuzamosak a térkép oldalaival, s mindegyiket a bal felső és a jobb alsó sarkának koordinátájával adjuk meg. Az egyes téglalapok átfedhetik egymást. (Minden koordináta 1 blokkot jelöl a térképen, azaz az (1,1) és (1,1) koordinátákkal megadott téglalap kerülete 4, területe 1 egység. A teljes térkép területe N*M, a kerülete pedig 2*N+2*M egység.)

Készíts programot **terkep.cbp (main.cpp)** a nagyított részek együttes kerületének és területének meghatározására!

Bemenet:

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



Kimenet:

A **terkep.ki** állomány egyetlen sorába a nagyított rész együttes kerületét és területét kell írni, egyetlen szóközzel elválasztva.

Példa:

```
terkep.be     terkep.ki
10 8 3        26 18
5 6 6 7
1 1 2 4
1 1 5 2
```

4. feladat: Jelek (25 pont)

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

Az **A** égitest esetén:

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

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

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

A **B** égitest esetén:

B1. A **01** egy észlelt jelsorozat.

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

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

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

Készíts programot **jelek.cbp (main.cpp)**, amely meghatározza, hogy a vizsgált jelsorozatok melyik égitestről érkezhettek!

Bemenet:

A **jelek.be** szöveges állomány első sorában a vizsgálandó jelsorozatok N ($1 \le N \le 100$) száma van. A következő N sor mindegyikében egy-egy jelsorozat van, amelynek hossza nem nagyobb, mint 255, és csak a 0 és az 1 karaktereket tartalmazhatja.

Kimenet:

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

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

B, ha a jelsorozat csak a **B** égitestről érkezhetett,

AB, ha a jelsorozat az **A** és a B égitestről egyaránt érkezhetett,

C, ha a jelsorozat sem az **A,** sem a **B** égitestről nem érkezhetett.

Példa:

```
jelek.be          jelek.ki
4                 A
0100              C
101011            B
000111            AB
010011
```