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 pontot (akkor sem, ha egyébként tökéletes lenne a megoldás); az objektív értékelés érdekében ugyanis a pontozóknak a programszövegekben egyetlen karaktert sem szabad javítaniuk, s az előre megadott javítási útmutatótól semmiben nem térhetnek 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 állományok neve minden esetben rögzített.

1. feladat: Fák (14 pont)

Egy fasorba N fát ültettek balról jobbra, egy vonalba. Mindegyik fának ismerjük a magasságát és a bal szélső fáról vett távolságát. Ha egy fát kivágunk, akkor az a jobboldali szomszédja felé dől, s amelyik szomszédját érinti, az is kidől.

Írj programot (*fak.pas* vagy *fak.c*), amely megadja, hogy minimum hány fát kell kivágnunk ahhoz, hogy az összes fa kidőljön, s melyik fa kivágása okozza a legtöbb fa kidőlését!

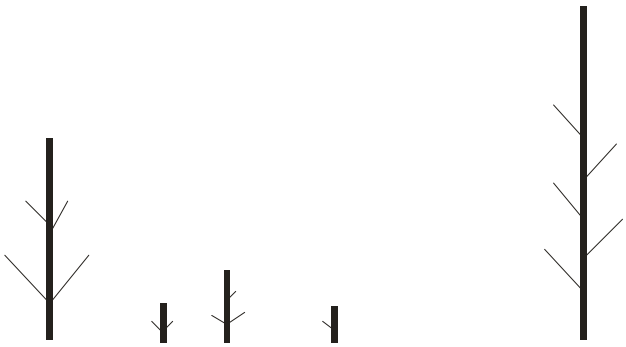
A *fak.be* állomány első sorában a fák N száma van (1≤N≤1000). A következő N sor mindegyike két számot, egy-egy fa leírását tartalmazza: a bal szélső fától vett T távolságát (1≤T≤1000) és a fa M magasságát (1≤M≤100). A fákat balról jobbra haladva adjuk meg.

A *fak.ki* állomány első sorába a minimálisan kivágandó fák számát kell írni, a második sorába pedig annak a kivágandó fának a sorszámát, amely kivágása esetén a legtöbb fa fog kidőlni.

Példa:

fak.be
```
5
0 6
3 1
5 2
8 1
15 10
```

fak.ki
```
3
1
```



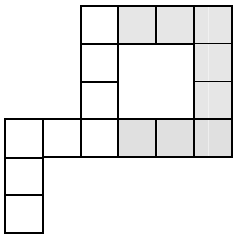
2. feladat: Pince (16 pont)

Pincét fúrnak: az E,J,B utasítások hatására egy egységnyi részt fúrnak, jobbra, illetve balra fordulnak 90 fokkal. A ()-ben levő részek a főágról leágazást jelentenek, előbb mindig a főágat fúrják végig, s utána kezdenek az elágazásokhoz. Szabályok: két ág nem érhet össze, sőt a sarkával sem találkozhat. Önmagában minden lépés biztosan olyan, hogy fúrni is kell, azaz teljesen nem mennek bele már kifúrt részbe (pl. nincs olyan sorozat, hogy EJJE).

Írj programot (*pince.pas* vagy *pince.c*), amely megadja hogy a pince helyes-e, s ha nem, akkor melyik lépésben találkoznak össze az ágak!

A *pince.be* állomány egyetlen sorában a pincét leíró karaktersorozat van (legfeljebb 10000 karakter a sor végéig). A pincefúrás közben az X- és az Y-koordináta nem lép ki a [-100..100] intervallumból. A kezdőkoordináta a (0,0).

A *pince.ki* állományba egyetlen számot tartalmazó sort kell írni: 0-t, ha a pince helyes, illetve az I számot, ha a pincét leíró karaktersorozatban az I-edik karakter hatására fúrt szakasznál találkoznak össze az ágak.



Példa:

pince.be	*pince.ki*
EEEJEEB(JEEEBEEEBEE)EEE	19

A példában az elágazás ásásának utolsó lépésében ütközünk a főág végébe, szürkítetten szerepel az elágazás.

3. feladat: Osztály (15 pont)

Egy osztályba N tanuló jár. Minden tanuló ismeri néhány osztálytársának telefonszámát.

Írj programot (*osztaly.pas* vagy *osztaly.c*), amely megadja azt a tanulót, akitől egy hír az ismert telefonokon keresztül továbbadva előbb-utóbb az osztály legtöbb tanulójához eljut!

Az *osztaly.be* állomány első sorában a tanulók N száma ($1 \le N \le 100$) van. A következő N sor mindegyike egy-egy tanuló által ismert telefonszámú tanulókat ír le, az állomány i+1-edik sorában azoknak a tanulóknak a sorszáma van, akiét az i-edik tanuló ismeri. Mindegyik sorban legfeljebb N-1 különböző egész szám van, egy-egy szóközzel elválasztva és 0-val zárva: az ismert telefonszámú tanulók sorszáma.

Az *osztaly.ki* állományba egyetlen sort kell írni, annak a tanulónak a sorszámát, akitől a legtöbb tanulóhoz eljuthat egy hír. Ha több ilyen tanuló van, akkor bármelyik sorszáma kiírható.

Példa:

osztaly.be	*osztaly.ki*
5 3 5 0 3 4 0 0 2 3 0 2 0	1

4. feladat: Vonat (15 pont)

Egy hosszú vasútvonal mentén ***N*** város helyezkedik el, minden városnak pontosan egy vasútállomása van a vonalon. Ismerjük a vonalon közlekedő vonatokat. Minden vonat adott i-edik városból indul és adott j-edik városba közlekedik (***i*** <***j***) és közben nem áll meg egyetlen közbülső állomáson sem. Az ***1***. városból indulva, vonattal közlekedve a lehető legtöbb várost szeretnénk meglátogatni.

Írj programot (*vonat.pas* vagy *vonat.c*), amely kiszámítja, hogy az ***1***. városból indulva mennyi a legtöbb meglátogatható város, és meg is ad egy útvonalat, amelyen haladva a legtöbb város meglátogatható!

Az *vonat.be* állomány első sorában két egész szám van, a városok ***N*** száma ($1 \le N \le 200$) és a járatok ***M*** ($1 \le M \le 3000$) száma. A további ***M*** sor mindegyike két egész számot tartalmaz (egy szóközzel elválasztva), az első szám ***i***, a járat indulási, a második szám ***j*** a járat érkezési állomása ($1 \le i < j \le N$). Az állomány ***i+1***-edik sora az ***i***-edik járat adatát tartalmazza.

Az *vonat.ki* állományba első sorába egyetlen egész számot kell írni, a legtöbb meglátogatható város ***K*** számát, beleértve az 1. induló várost is! A második sor pontosan ***K-1*** számot tartalmazzon (egy-egy szóközzel elválasztva), a járatok bemenetbeli sorszámait az utazás sorrendjében. Több megoldás esetén bármelyik kiírható.

Példa:

vonat.be

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

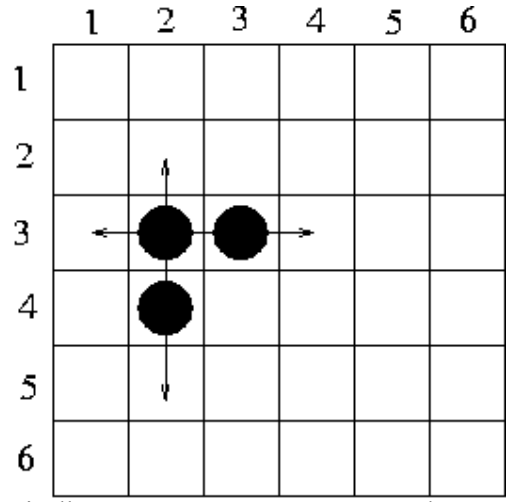
vonat.ki

```
5
1 5 7 6
```

5. feladat: Játék (15 pont)

Tekintsük a Solitaire játéknak azt a változatát, amelyet ***6x6***-os négyzetrácsos táblán lehet játszani. A táblára három fekete korongot helyeznek három különböző mezőre, ez a kezdeti játékállás. A játék során minden lépésben egy korongot lehet mozgatni az alábbi szabály szerint.

- Csak üres mezőre lehet lépni.
- A négy szomszédos mező valamelyikére lehet lépni, balra, jobbra, felfelé vagy lefelé.
- Ha a lépés irányába eső szomszédos mezőn van korong, akkor azt az egy korongot át lehet lépni.



A a (3,2) mezőn álló korong négy lehetséges lépése: (2,2), (3,1), (5,2), (3,4), mint az ábrán látható.

Írj programot (*jatek.pas* vagy *jatek.c*), amely kiszámítja, hogy adott kezdeti játékállásból legkevesebb hány lépés végrehajtásával lehet eljutni adott végállásba!

Az *jatek.be* két sort tartalmaz, az első sor a kezdeti játékállást, a második pedig a végállást írja le. Mindkét sor 6 egész számot tartalmaz, a három korong koordinátáit. Az i-edik (i=1,2,3), számpár az i-edik korong sor, illetve oszlopkoordinátáját jelenti. A sorokat fentről lefelé, az oszlopokat balról jobbra sorszámozzuk 1-től 6-ig. A három korong sorrendje közömbös a végállásban!

Az *jatek.ki* állomány első és egyetlen sora egy egész számot tartalmazzon, azon legkevesebb lépések számát, amennyi lépéssel el lehet jutni a kezdeti játékállásból a végállásba!

Példa:

jatek.be

```
3 2 3 3 4 2
2 3 3 3 3 4
```

jatek.ki

```
3
```

Elérhető összpontszám: 75 pont + 25 pont az 1. fordulóból