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: Kamion (50 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.

Írj programot (projekt név: kamion (kamion.cbp; main.cpp)), amely kiszámítja azt a legkisebb kamion kapacitást, amellyel a szállítás megoldható, ha minden kamion azonos kapacitású!

A „kamion.be” állomány első sorában a városok N ($1<N\leq200$) száma van A központi raktár az 1. városban van, és onnan nem kell szállítani. A következő N-1 sor 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 program a „kamion.ki” állományba azt a legkisebb kamion kapacitást (egész szám) írja ki, amekkora kapacitású kamionokkal az összes alkatrész elszállítható!

Példa:

kamion.be	kamion.ki
8	12
1 2	
1 3	
1 4	
2 5	
3 6	
3 7	
4 8	

2. feladat: Akadálypálya (50 pont)

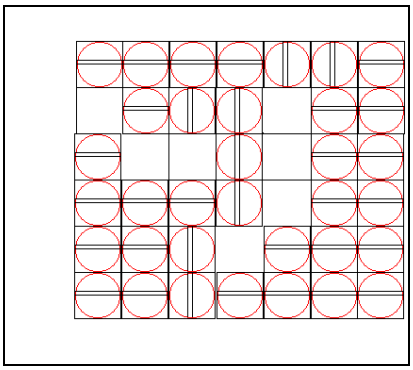
Egy jármű négyzetrácsos elrendezésű pályaelemekből álló, M sorból, N oszlopból álló pályán mozoghat. Minden pályaelem vagy üres, vagy a közepén áthaladó sínt tartalmaz, amelyen a jármű haladhat. Egy pályaelem négy szomszédja a négyzetrácsos elrendezésben a tőle balra, jobbra, lefelé vagy fölfelé lévő 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 (projekt név: palya (palya.cbp; main.cpp)), amely kiszámítja 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 „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:

```
palya.be:          palya.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
```

3. feladat: Terv (50 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 (projekt név: terv (terv.cbp; main.cpp)), 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 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:

```
terv.be:           terv.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: Elfogó (50 pont)

A városi rendőrség egy veszélyes bűnöző elfogását tervezi, aki gépkocsival folyamatosan közlekedik a város utcáin. A rendőrségnek korlátozottak az 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 (projekt név: elfogo (elfogo.cbp; main.cpp)), 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≤N≤200)*** száma, a második sorában az utcák ***(1≤M≤10000)*** száma van. A következő ***M*** sor mindegyike egy ***A B*** számpárt ***(1≤A, B≤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
```