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: Folyók (25 pont)

Magyarország folyóiról feljegyeztük, hogy milyen másik folyóba folynak bele (pl. a Rába a Dunába, a Sajó a Tiszába, a Hernád a Sajóba, …). Minden folyó legfeljebb 1 másikba folyhat bele, de lehet hogy egybe sem (pl. Duna, de a Zala sem folyóba folyik bele).

A folyók szennyezettségének megállapításához szükség van arra, hogy megállapítsuk, melyik folyóba honnan érkezhet *szennyezett* víz.

Készíts programot **`folyo.cbp (main.cpp)`**, amely megadja, hogy

A. egy folyóba mely folyókból érkezhet víz (akár másik folyón keresztül is);

B. egy folyóba került szennyezés mely más folyókat szennyezhet meg!

A folyo.be állomány első sorában az az N egész szám van, ahány folyóról tudjuk, hogy melyikbe folyik bele ($1 \le N \le 1000$). A következő 2*N sor mindegyike egy-egy folyó nevét tartalmazza, a második, negyedik, ... sor azt hogy melyik folyó, a hamadik, ötödik, … pedig azt, hogy melyikbe folyik bele. Az utolsó sorban egy folyó neve van, az, amelyről az A és a B kérdés szól. (A folyónevek legfeljebb 20 betűsek.)

A folyo.ki állomány első sorába azoknak a folyóknak a K számát kell írni, amelyekből érkezhet víz a folyo.be állomány utolsó sorában levő folyóba. A következő K sorba kell kiírni soronként a folyók nevét. A K+1-edik azoknak a folyóknak az L számát kell írni, amelyekbe eljuthat a szennyeződés a megadott folyóból! Az ezt követő L sorba kell kiírni soronként a folyók nevét.

Példa:

```
folyo.be
4
Zagyva
Tisza
Sajó
Tisza
Tarna
Zagyva
Gyöngyös
Tarna
Zagyva
```

```
folyo.ki
2
Tarna
Gyöngyös
1
Tisza
```

2. feladat: Kemence (25 pont)

A porcelángyár égetőkemencéjéhez futószalagon érkeznek az égetésre váró tárgyak. Minden tárgynak ismert a súlya és a kiégetéséhez minimálisan szükséges idő. Az égetésre váró tárgyakat az érkezésük sorrendjében kell kiégetni. Egyszerre több tárgyat is rakhatunk a kemencébe, az összsúlyuk azonban nem haladhatja meg a kemence kapacitását. Az égetési idő egy menetben mindig a kemencébe rakott tárgyak minimális égetési idejének a maximuma kell legyen.

Készíts olyan programot **kemence.cbp (main.cpp)**, amely kiszámítja, hogy legkevesebb mennyi idő kell az összes tárgy kiégetéséhez, továbbá megadja azt is, hogy ezen idő eléréséhez mely tárgyakat kell egy-egy menetben a kemencében együtt égetni.

A kemence.be állomány első sora két egész számot tartalmaz; a tárgyak N (1≤N≤10 000) számát és a kemence K (1≤K≤1000) kapacitását. A következő N sor mindegyike két egész számot tartalmaz; egy tárgy S (1≤S≤1000) súlyát és E (1≤E≤100) minimális égetési idejét.

A kemence.ki állomány első sorába az összes tárgy kiégetéséhez minimálisan szükséges időt kell írni. A következő sorok mindegyikébe két egész számot, I-t és J-t kell írni egy szóközzel elválasztva, I az első, J pedig az utolsó tárgy sorszáma, amelyek egyszerre kerülnek a kemencébe.

Példa:

kemence.be
```
6 10
3 10
2 12
7 20
5 15
3 11
2 16
```

kemence.ki
```
46
1 1
2 3
4 6
```

3. feladat: Terem (25 pont)

Iskolád alapításának évfordulóján nagyszabású ünnepséget szervez. Egy napra sok eseményt tervez. Kiderült, hogy lesznek események, amelyek részben egy időben zajlanak. Ezért meg kell határozni, hogy legkevesebb hány termet kell előkészíteni ahhoz, hogy minden esemény számára legyen terem foglalva, és természetesen az események ne ütközzenek. Minden betervezett eseménynek ismerjük a kezdési és befejezési időpontját, amit percben adtak meg. Ha egy esemény az A perctől a B percig tart, akkor ugyanabba a terembe beosztott bármely másik esemény vagy A-nál korábban véget ér, vagy B-nél később kezdődhet.

Készíts programot **terem.cbp (main.cpp)**, amely kiszámítja, hogy legkevesebb hány terem kell ahhoz, hogy minden betervezett eseményt meg lehessen tartani, továbbá megad egy lehetséges terembeosztást.

A terem.be állomány első sorában az események N száma van (1≤N≤1000). A következő N sor mindegyikében két egész szám, A és B van egy szóközzel elválasztva. Az A szám egy esemény kezdő, a B pedig ugyanezen esemény befejező időpontja (1≤A≤B≤ 1440).

A terem.ki állomány első sorába az összes esemény beosztásához szükséges legkevesebb terem T számát kell írni! A következő T sorban kell megadni a termek beosztását. Egy sorba azon események sorszámait kell írni egy-egy szóközzel elválasztva, amelyek ugyanazon teremben lesznek megtartva (a különböző sorokban lévők pedig mind külön teremben).

Példa:

terem.be
```
8
1100 1200
500 520
510 570
600 630
630 700
700 800
600 800
650 700
```

terem.ki
```
4
1 2 4 8
3 5
6
7
```

4. feladat: Piramis (25 pont)

A piramisépítők egy négyzet alapú területre építik a piramist. A terület minden egységnyi négyzetére adott darabszámú kőkockát helyeznek. Amikor egy újabb követ kell elhelyezni, akkor valahonnan a piramis széléről indulnak, és úgy haladnak, hogy minden lépésben pontosan eggyel magasabb szomszéd helyre lépnek. (A szomszédos hely átlósan nem lehet!) A követ mindig oda teszik, ahonnan nem tudnak szomszédos helyre továbblépni.

Készíts programot **`piramis.cbp (main.cpp)`**, amely megadja a leghosszabb utat, amelyen a piramisépítők egy követ elvihetnek a helyére!

A piramis.be állomány első sorában a piramist tartalmazó négyzet oldalhossza van ($1 \le N \le 100$). A következő N sor mindegyike N egész számot tartalmaz, egy-egy szóközzel elválasztva, az egyes pozíciókban levő kőkockák számát ($0 \le darabszám \le 10000$).

A. piramis ki állomány első sorába a leghosszabb út hosszát kell írni (azon lépések számát, ahány lépés alatt egy tetszőleges kezdőpozícióból szomszéd helyeken át egyesével lehet felfelé lépkedni), a második sorba pedig az ehhez az úthoz tartozó kezdő pozíció sor- és oszlopindexét. Ha sehonnan sem lehet lépni, akkor az első sorba 0, a második sorba tetszőleges pozíció írandó.

Példa:

`piramis.be`

```
6
1 2 2 2 2 2
4 3 4 2 2 1
1 1 5 6 1 8
1 1 1 9 6 7
1 3 4 4 5 1
1 2 1 1 1 1
```

`piramis.ki`

```
5
1 1
```

Itt a bal felső sarokból indulva pontosan 5 lépést lehet megtenni. Más helyről indulva ennél csak kevesebbet lehet.