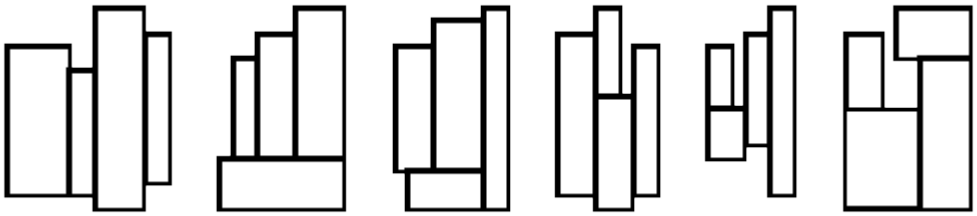
Kedves Versenyző! A megoldások értékelésénél csak **a programok futási eredményeit** vesszük tekintetbe. Ezért igen fontos a **specifikáció pontos betartása**. A programokat csak a feladatkiírásban leírt szabályoknak megfelelő adatokkal próbáljuk ki, emiatt nem kell ellenőrizni, hogy a bemenő adatok helyesek-e, illetve a szükséges állományok léteznek-e. Ha a programnak valamilyen állományra van szüksége, akkor azt mindig az aktuális könyvtárba kell rakni. Az állomány neve minden esetben rögzített.

1. feladat: Téglalap (50 pont)

Adott négy téglalap. Helyezzük egymás mellé a téglalapokat, átfedés nélkül úgy, hogy a befoglaló téglalap a lehető legkisebb területű legyen. A négy téglalapot úgy kell elhelyezni, hogy az oldalaik párhuzamosak legyenek a befoglaló téglalap megfelelő oldalaival. Az ábra a téglalapok 6 lehetséges egymás mellé helyezését mutatja. Ezek alapesetnek tekinthetők, mert minden egyéb elrendezés előállítható ezekből forgatással vagy tükrözéssel.



Készíts programot, amely előállítja a legkisebb befoglaló téglalapot! Ha több, azonos területű, de különböző oldalhosszúságú befoglaló téglalap is lehet, akkor mindegyiket elő kell állítani!

A `teglalap.be` állomány négy sorból áll. Minden sorban két egész szám van: az egyes téglalapok oldalainak hossza. Az oldalak hossza legalább 1 és legfeljebb 50.

A `teglalap.ki` állomány első sorába a legkisebb befoglaló téglalap területe kerüljön! A következő sorokban egy-egy megoldás legyen. Minden megoldást egy (p,q) számpár ír le, ahol $p \leq q$. A sorokat p szerint növekvő sorrendben kell kiírni a fájlba. Minden sornak különbözőnek kell lennie.

Példa:

teglalap.be	teglalap.ki
1 2	40
2 3	4 10
3 4	5 8
4 5	

2. feladat: Vásárlás (50 pont)

Egy boltban minden terméknek adott ára van. Például egy szál virág 2 ICU-ba, egy váza 5 ICU-ba kerül (ICU=Informatics Currency Unit). A forgalom növelésére a bolt különféle árkedvezményeket ad. A kedvezményes ajánlat egy vagy több termékre vonatkozik. Pl. 3 szál virág ára 6 helyett 5 ICU, 2 váza és 1 virág ára 12 helyett 10 ICU.

Írj programot, amely kiszámítja a vásárolt (kosárnyi) áruk árát, kihasználva a kedvezményes ajánlatokat úgy, hogy a fizetendő összeg a lehető legkisebb legyen! A kosárban levőnél több áruval nem számolhatunk akkor sem, ha ezzel a fizetendő végösszeg kisebb lenne.

Például a fenti árkedvezmények mellett két váza és három virág lehető legalacsonyabb ára 14 ICU, ugyanis két váza és egy virág kedvezményes ára 10 ICU, 2 virág alapára pedig 4 ICU.

A `vasarlas.be` első sora egy b szám, amely megadja, hogy a kosárban hány (különböző) fajta árucikk található (0≤b≤5). A következő b sor mindegyikében három szám van: c, k és p. A c az árucikk egyedi kódja (1≤c≤999). A k az adott árucikkből vásárolt áruk darabszáma (1≤k≤5). A p az árucikk kedvezmény nélküli egységára (1≤p≤999). A következő sorban a kedvezményes ajánlatok s száma van (0≤s≤99). A következő sorok egy-egy kedvezményes ajánlatot írnak le. Minden sorban az első szám (n) megadja, hogy a kedvezményes ajánlat hány különböző árucikkből áll. A következő n számpár (c,k) azt jelenti, hogy a c kódú árucikkből (1≤c≤999) az ajánlatban pontosan k db (1≤k≤5) van. Az utolsó szám (p) a kedvezményes ár (1≤p≤9999). Ez mindig kisebb, mint a kedvezmény nélküli árak összege.

A `vasarlas.ki` állományba egyetlen számot kell írni, a bemenetben leírt termékekért fizetendő, lehető legkisebb összeget.

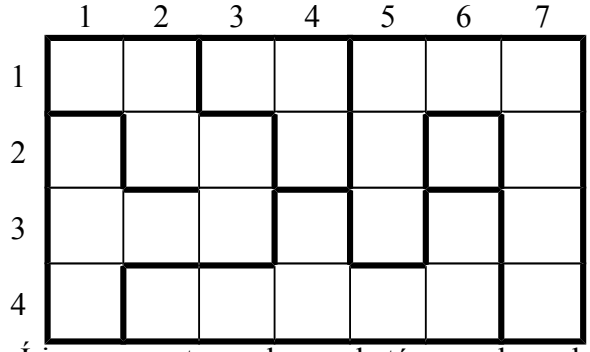
Példa:

`vasarlas.be`	`vasarlas.ki`

`2`
`7 3 2`
`8 2 5`
`2`
`1 7 3 5`
`2 7 1 8 2 10`
	`14`

3. feladat: Kastély (50 pont)

A következő ábra egy kastély alaprajzát mutatja.



Írj programot, amely meghatározza, hogy hány szoba van a kastélyban; mekkora a legnagyobb szoba; illetve melyik falat kell kivenni ahhoz, hogy olyan nagy szobát hozzunk létre, amekkorát csak lehetséges!

A `kastely.be` állomány állományban az első szám az alaprajz sorainak (1≤M≤50), a második pedig az oszlopainak (1≤N≤50) a számát adja meg. Az ezután következő sorokban egy-egy négyzetet írnak le a számok (0≤p≤15). Ez a szám a következő kódok összegeként áll elő: 1 (nyugati fal), 2 (északi fal), 4 (keleti fal), 8 (déli fal). A belső falakat kétszer adjuk meg, pl. az (1,1) négyzet déli fala a (2,1) négyzet északi falaként is szerepel. A kastélyban legalább 2 szoba van.

A `kastely.ki` állományba három sort kell írni. Az első sorba a szobák számát, a másodikba a legnagyobb szoba területét (a négyzetek számával kifejezve), végül a harmadik sorba az eltávolítandó fal leírását (először az eltávolítandó fal melletti négyzet sorának, azután az oszlopának a számát kell megadni, majd pedig azt, hogy ebben a négyzetben melyik égtáj felé esik az eltávolítandó fal – E,K,D,N). Több megoldás esetén a harmadik sorba bármelyik kiírható.

Példa:

`kastely.be`	`kastely.ki`

```
4                                   5
7                                   9
11 6 11 6 3 10 6                    4 1 K
7 9 6 13 5 15 5
1 10 12 7 13 7 5
13 11 10 8 10 12 13
```

4. feladat: Prímek (50 pont)

Az ábrán egy négyzet látható. Minden sora, minden oszlopa és a két főátlója egy-egy ötjegyű prímszámnak olvasható. A sorokat balról jobbra, az oszlopokat fölülről lefelé, a két főátlót pedig balról jobbra kell kiolvasni.

1	1	3	5	1
3	3	2	0	3
3	0	3	2	3
1	4	0	3	3
3	3	3	1	1

A prímszámokban a számjegyek összegének ugyanannak kell lenni (ez a példában 11)! A bal felső sarokban levő számjegyet előre megadjuk (a példában ez a szám 1). Ugyanaz a prímszám egynél többször is felhasználható ugyanabban a négyzetben. Ha több megoldás van, akkor mind meg kell adni! A prímszámok nem kezdődhetnek nullákkal, azaz pl. a 00003 nem ötjegyű prímszám!

Készíts programot, amely ilyen négyzeteket készít!

A `primek.be` állomány első sora a prímszámok számjegyeinek összegét, a második pedig a bal felső sarokba írandó számjegyet tartalmazza.

A `primek.ki` állományba minden megoldást 5 sorba kell kiírni! Minden sor egy-egy ötjegyű prímszámot tartalmazzon! A megoldások között egy-egy üres sor legyen!

Példa:

`primek.be`

```
11
1
```

`primek.ki`

```
11351
14033
30323
53201
13313

11351
33203
30323
14003
33311

13313
13043
32303
50231
13331
```