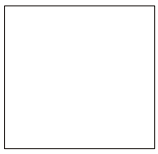
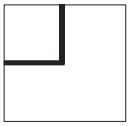
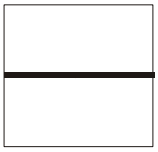
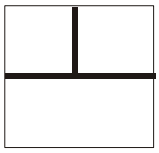
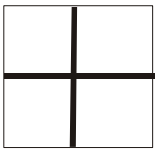
Válogatás a Nemes Tihamér Verseny feladataiból

1. feladat: Villamos (2002. évi 3. forduló, 3. korcsoport, 1. feladat)

Egy négyzetrácsos szerkezetű városban villamosok közlekednek. A villamos pályákat egy térképpel írjuk le, ahol minden térképelem egy-egy négyzetet jelent. A következő térképelemeket használhatjuk:

0:  1:  2: 

3:  4: 

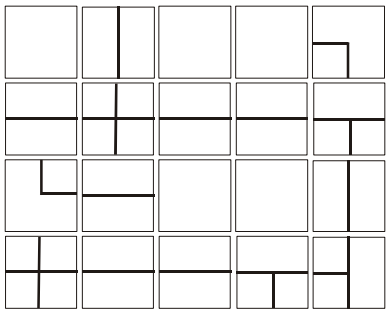
Minden térképelem négyféle állásban rajzolható:

0: az ábrákon látható 1: 90 fokkal az óramutató járása szerint elforgatva

2: 180 fokkal elforgatva 3: 270 fokkal az óramutató járása szerint elforgatva

Készíts programot (VILLAMOS.PAS vagy VILLAMOS.C), amely megadja, hogy egy adott pozícióról egy másikra a villamos minimum hány lépés alatt juthat el, illetve minimum hány lépés alatt juthat el akkor, ha azt az elemet, amin éppen áll, el tudja forgatni balra vagy jobbra 90 fokkal (ekkor azonban a forgatás is lépésnek számít)!

A VILLAMOS.BE állomány első sorában térkép sorai ($1 \le N \le 100$) és oszlopai ($1 \le M \le 100$) száma van. A következő N sorban soronként M számjegy-pár található, egy-egy szóközzel elválasztva, mindegyikben a térkép egy sorának leírása. A térkép kód-párokkal adható meg: minden helyet az ábrasorszám és az elforgatás kódja azonosít.



```
00 21 00 00 13
20 40 20 20 32
11 20 00 00 21
40 20 20 32 31
```

Az utolsó sorba 4 egész számot kell írni a kezdőpozíció sor- és oszlopindexét, valamint a célpozíció sor- és oszlopindexét.

A VILLAMOS.KI állomány első sorába azt a minimális lépésszámot kell írni, ami alatt a villamos eljuthat a kezdőpozícióról a végpozícióra; a második sorba pedig ugyanezt a számot abban az esetben, ha a villamos forgathatja azt az elemet, amin éppen áll!

Példa:

VILLAMOS.BE

```
4 5
00 21 00 00 13
20 40 20 20 32
11 20 00 00 21
40 20 20 32 31
1 2 4 1
```

VILLAMOS.KI

```
10
6
```

Megjegyzés:

Út az 1. esetben: (1,2),(2,2),(2,3),(2,4),(2,5),(3,5),(4,5),(4,4),(4,3),(4,2),(4,1)

Út a 2. esetben: (1,2),(2,2),(2,1),fordít,(3,1),fordít,(4,1)

2. feladat: Mozgat (2002. évi 3. forduló, 3. korcsoport, 2. feladat)

Minden szövegszerkesztővel végezhető kivágás-beillesztés művelet. Egy N számú sort tartalmazó dokumentumon K kivágás-beillesztés műveletet végeztek, amelyeket ismerünk. Minden művelet egy A, B, C számhármassal van megadva; amely azt jelenti, hogy a dokumentum A-tól B-ig terjedő sorait (A-t és B-t is beleértve) átmásolták a B-edik sor után és kitörölték a beillesztett sorokat az eredeti helyéről.

Írj programot (`MOZGAT.PAS, MOZGAT.C, MOZGAT.CPP`), amely kiszámítja, hogy

A. a dokumentum első 10 sora hova került a műveletek elvégzése után;

B. az eredeti dokumentum mely sorai kerültek az első 10 sorba a műveletek hatására.

A `MOZGAT.BE` állomány első sora két egész számot tartalmaz: N a dokumentum sorainak száma ($1 \le N \le 1000000$), K pedig a műveletek száma ($1 \le K \le 1000$). A további K sor mindegyikében három egész szám van: A B C, ami egy művelet leírása. A számokra teljesülnek a következő egyenlőtlenségek: $1 \le A \le B \le N$ és $0 \le C < A$ vagy $B \le C \le N$. Ha C=0, akkor a beillesztés az első sor elé kerül.

A `MOZGAT.KI` állományba két sort kell írni, mindkét sorban 10 db szám legyen egy-egy szóközzel elválasztva. Az első sor sorrendben azon sorok számát tartalmazza, ahová az eredeti dokumentum első 10 sora került a műveletek hatására. A második sor rendre azokat az eredeti dokumentum sor számait tartalmazza, amelyek az első 10 sorba kerültek a műveletek hatására.

3. feladat: Ütemezés (2002. évi 3. forduló, 3. korcsoport, 3. feladat)

Mekk Elek ezermester nagyon népszerű vállalkozó, sokan keresik meg megbízásokkal. A mester a következő évre beérkezett igényeket kívánja ütemezni. Minden munkát egy nap alatt tud elvégezni. Minden igényelt munkának ismeri a határidejét, ami azt jelenti, hogy ha elvállalja a munkát, akkor a határidejéig el kell végeznie.

Írj programot (UTEMEZ.PAS, UTEMEZ.C, UTEMEZ.CPP), amely kiszámítja a munkák egy legnagyobb elemszámú olyan részhalmazát, hogy a kiválasztott munkákat be lehet úgy osztani, hogy mindegyik elvégzésre kerül a határidejéig. A programodnak egy ilyen beosztást is meg kell adnia.

Az UTEMEZ.BE állomány első sora a munkák N számát ($1 \le N \le 10000$) tartalmazza. A következő sor mindegyike egy H pozitív egész számot tartalmaz ($1 \le H \le 365$), ami egy munka határideje. A munkákat a bemenet-beli sorszámukkal azonosítjuk.

Az UTEMEZ.KI állomány első sorában a kiválasztott munkák M száma legyen. A következő M sor mindegyikébe két számot kell írni egy-egy szóközzel elválasztva. Az első szám egy kiválasztott munka sorszáma, a második pedig annak a napnak a sorszáma, amelyikre ezt a munkát ütemeztük.

4. feladat: Tükörszó (2002. évi 3. forduló, 3. korcsoport, 4. feladat)

Egy szót tükörszónak nevezünk, ha balról és jobbról olvasva megegyezik. (Tehát minden egybetűs szó tükörszó.) Minden szóban található tükörszó, amin azt értjük, hogy ha kitörlünk belőle betűket, akkor tükörszót kapunk.

Írj programot (`TUKOR.PAS, TUKOR.C, TUKOR.CPP`), amely kiszámítja egy adott szóban található leghosszabb tükörszó hosszát!

A `TUKOR.BE` állomány első és egyetlen sorában egy legfeljebb 100 karaktert tartalmazó ***S*** szó van.

A `TUKOR.KI` állományba az ***S*** szóban található leghosszabb tükörszó hosszát kell írni.

5. feladat: Régió (2003. évi 3. forduló, 3. korcsoport, 1. feladat)

Egy megyén belül a településeket régiókba szeretnék csoportosítani. Ismerjük az egyes települések koordinátáit. Két település távolságán a koordináta-különbségeik abszolút értékének összegét értjük, azaz TÁVOLSÁG((x,y),(a,b))=|x-a|+|y-b|.

Két települést azonos régióba teszünk, ha vezet egyikből a másikba olyan út, amely a régió településein halad át és az úton egymást követő települések távolsága legfeljebb T kilométer.

Írj programot (REGIO.PAS, REGIO.C,…), amely megadja, hogy a települések hány régiót alkotnak, és mely települések tartoznak egy régióba!

A REGIO.BE szöveges állomány első sorában a városok N száma (2≤N≤100) és a régióba kerülés határát jelentő T távolság (1≤T≤100) van. A következő N sor mindegyikében egy-egy számpár van, az adott város x- és y-koordinátája (0 és 1000 közötti egész számok), egy szóközzel elválasztva.

A REGIO.KI szöveges állomány első sorába a legkisebb K számot kell írni, ahány régióba lehet besorolni a településeket. A következő K sorba az egyes régiókat kell írni, tetszőleges sorrendben. Egy sorba a régióba tartozó települések sorszámát kell írni, egy-egy szóközzel elválasztva, növekvő sorrendben.

Példa:

`REGIO.BE`

```
6 50
100 100
100 220
100 120
300 100
120 140
310 90
```

`REGIO.KI`

```
3
1 3 5
2
4 6
```

6. feladat: Repülőút (2003. évi 3. forduló, 3. korcsoport, 2. feladat)

Egy repülőtársaság N város között üzemeltet járatokat. A városokat a természetes számokkal azonosítják 1-től N-ig. A társaság jelentős kedvezményt ad, ha az utas olyan útvonalat választ, hogy az utazás során mindig nagyobb sorszámú városba megy. Az A városból szeretnénk eljutni a B városba (A<B) kedvezményes útvonalon.

Írj programot (REPUT.PAS, REPUT.C, …), amely megadja azokat a városokat, amelyeken mindenképpen át kell haladnunk, valamint azokat a város-párokat, amelyek közötti járatot mindenképpen igénybe kell venni bármely kedvezményes útvonalon akarunk az A városból a B városba jutni!

A REPUT.BE szöveges állomány első sorában a városok N száma (1≤N≤100) és a járatok M száma (1≤M≤1000) van. A következő M sor mindegyikében egy-egy P Q számpár (1≤P<Q≤N) van, egy szóközzel elválasztva. Ez azt jelenti, hogy van járat az P és a Q város között. Az utolsó sorban az A és a B város sorszáma van, egy szóközzel elválasztva (1≤A<B≤N).

A REPUT.KI szöveges állomány első sorába a kikerülhetetlen városok K számát kell írni, majd ettől egy-egy szóközzel elválasztva a kikerülhetetlen városok sorszámát növekvő sorrendben. A felsorolásban szerepeljen az indulási P, és az érkezési Q város is. A második sorba a kikerülhetetlen járatok M számát kell írni. A következő M sor mindegyikébe egy-egy elkerülhetetlen járatot kell írni, a két város sorszámát egy szóközzel elválasztva.

Példa:

REPUT.BE	REPUT.KI
12 14 1 2 1 3 2 4 3 4 4 5 5 6 5 7 6 8 7 8 8 9 9 10 8 10 3 11 7 12 1 10	3 4 5 8 1 4 5

7. feladat: Titkos társaság (2003. évi 3. forduló, 3. korcsoport, 3. feladat)

Egy titkos társaság hierarchikusan épül fel, minden tagja csak a felettesét és a hozzá közvetlenül beosztott legfeljebb két tagot ismeri. A társaságnak pontosan egy olyan tagja van, akinek nincs főnöke. Bármelyik tag küldhet levelet bármelyik tagnak. Azonban minden levél csak úgy juthat el a feladótól a címzetthez, hogy egy lépésben vagy a közvetlen főnökhöz, vagy közvetlen beosztotthoz továbbítódik.

Írj programot (TARSASAG.PAS, TARSASAG.C, …), amely adott két, X és Y tagra kiszámítja az alábbi kérdésekre adandó választ!

A. Hány beosztottja –nem csak közvetlen– van az X és a Y tagnak?

B. Hány lépéssel továbbítódik egy levél, ha X küld levelet Y-nek?

C. Mennyi a legkevesebb lépésszám, ami alatt biztosan odaér egy levél bárki legyen is a feladó, illetve a címzett?

A TARSASAG.BE szöveges állomány első sorában a társaság tagjainak N száma ($1 \le N \le 1000$), valamint a két tag X és Y sorszáma ($1 \le X,Y \le N$) van. A társaság tagjai olyan sorszámot kaptak 1 és N között, hogy mindenkinek nagyobb a sorszáma, mint a közvetlen főnökéé. A második sor pontosan N db 0 és N közötti egész számot tartalmaz egy-egy szóközzel elválasztva. Az i-edik szám a társaság i-sorszámú tagjának közvetlen főnökét adja. A sorban az első szám 0, mivel pontosan egy tagnak, az 1 sorszámúnak nincs főnöke.

A TARSASAG.KI szöveges állomány első sorába az A, második sorába a B, harmadik sorába a C kérdésre adott választ kell írni.

Az első sorba két számot kell írni, az X és a Y tag beosztottjainak a számát. A második sorba egyetlen számot kell írni, azt a lépésszámot, amely alatt egy levél eljut az X tagtól a Y taghoz. A harmadik sorba egyetlen számot kell írni, ami a legkevesebb lépésszám, ami alatt biztosan odaér egy levél bárki legyen is a feladó, illetve a címzett.

Példa:

TARSASAG:BE	TARSASAG.KI
7 2 5 0 1 1 3 3 5 5	0 2 3 4

8. feladat: Rendőrök (2003. évi 3. forduló, 3. korcsoport, 4. feladat)

Egy autópálya mentén N város helyezkedik el. Bizonyos városokban autópálya rendőrök tartózkodnak, némelyikben több is, némelyikben egy sem. Összesen legfeljebb N rendőr van. Azt szeretnénk elérni, hogy a lehető legtöbb városban legyen rendőr, ezért át kell csoportosítani. Az átcsoportosítást a lehető legkisebb összköltséggel kell végrehajtani. Egy rendőr i. városból a j.-be történő átmozgatásának költsége a várossorszámok különbségének abszolút értéke: |i-j|.

Írj programot (RENDOR.PAS, RENDOR.C, …), amely kiszámítja az átcsoportosítás lehető legkisebb összköltségét és megadja azt, hogy az átcsoportosítás után mely városokban lesz rendőr!

A RENDOR.BE szöveges állomány első sorában a városok N száma ($1 \le N \le 100$) van. A második sorban pontosan N szám van egy-egy szóközzel elválasztva. Az i-edik szám azt adja meg, hogy az i-edik városban kezdetben hány rendőr tartózkodik. Összesen legfeljebb N rendőr van a városokban.

A RENDOR.KI szöveges állomány első sorába azt a legkisebb összköltséget kell írni, amellyel elérhető, hogy a legtöbb városban legyen rendőr. A második sorba pontosan N számot kell írni, az i-edik szám 1-es legyen, ha az i-edik városban lesz rendőr az átmozgatás után, egyébként 0.

Példa:

RENDOR.BE

```
7
0 1 0 3 2 0 0
```

RENDOR.KI

```
5
1 1 1 1 1 1 0
```