Kedves Versenyző! A megoldások értékelésénél csak **a programok futási eredményét** 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: Térkép (50 pont)

Egy térképet egy NxM-es mátrixban ábrázolunk. A városokat a mátrixban a 2-es számjegy, az utakat pedig az 1-es számjegy jelzi. A többi ponthoz tartozó érték 0. Az utak minden pontból a 4 szomszédos pont irányában folytatódhatnak, azaz átlósan lépni nem lehet. Egy város több 2-es értékű pontból is állhat, de két város sehol sem érintkezhet egymással, azaz nincs szomszédos pontjuk.

Készíts programot (Projekt név: terkep (terkep.cbp; main.cpp)), amely két város esetén megadja a városok területét (hány 2-es értékű pontból áll), valamint a közöttük vezető legrövidebb út hosszát az alábbi háromféle módon:

A. a legrövidebb út az az út, ami a legkevesebb várost érint a kiindulásin kívül; a hossza pedig az érintett városok szám

B. a legrövidebb út az az út, ami legkevesebb, egyik városhoz sem tartozó (1-es értékű) közbülső ponton halad át, a hossza pedig ezen pontok száma;

C. a legrövidebb út az az út, amin a leggyorsabban el lehet érni a másik városba, feltételezve, hogy városban feleakkora a sebesség, mint a városok közötti utakon, azaz az 1-essel jelölt pontot 1, a 2-essel jelölt pontot pedig 2 időegység alatt lehet elhagyni; s a hossza az út megtételéhez szükséges idő.

A terkep.be állomány első sorában N és M értéke van ($1 \le N,M \le 100$), egy-egy szóközzel elválasztva. A további N sor mindegyike pontosan M számjegyet tartalmaz (csak 0, 1 vagy 2 lehet) szóközök nélkül, a térkép egyes sorai leírását. Az állomány utolsó sora két város indexeit ((X,Y) és (U,V)), azaz négy számot tartalmaz ($1 \le X,U \le N$, $1 \le Y,V \le M$), s a feladat az (X,Y)-ból (U,V)-be vezető legrövidebb út megtalálása. Az első sor első (bal oldali) elemének koordinátái (1,1), az N. sor M. elemének pedig (N, M).

A terkep.ki állományba négy sort kell írni! Az első sorba az (X,Y), illetve az (U,V) pontot tartalmazó város területét kell írni, egy szóközzel elválasztva! A következő három sorban egyetlen szám szerepeljen: az A, B, illetve C feltétel szerinti legrövidebb út hossza! Ha nincs út a két város között, akkor -1-et kell írni a második, a harmadik és a negyedik sorba!

Példa:

terkep.be
```
9 10
2200000000
0200000000
0110000000
0010000000
0011200000
0001220000
0000211122
0000000122
0000000112
1 1 9 10
```

terkep.ki
```
3 5
2
8
22
```

2. feladat: Karaván (50 pont)

Egy sivatagban N város található, melyek között az év egy időszakában naponta indulnak karaván. Ezen az időszakon kívül azonban egyetlen sem indul.

Készíts programot (Projekt név: karavan (karavan.cbp; main.cpp)), amely meghatározza két adott, A B városra és H határidőre a következőket:

A. Legkorábban mikorra érhetünk az A városból a B városba;

B. Legalább hány nap kell ahhoz, hogy az A városból a B városba érjünk, ha nincs megkötés sem az idulási, sem az érkezési időre;

C. Legkésőbb melyik napon kell elindulni az A városból, hogy legkésőbb a megadott H határidőig a B városba érjünk.

A karavan.be állomány első sorában a városok száma (1≤N≤100) és a karaván-kapcsolatok száma (1≤M≤5000) van. A következő M sor mindegyikében két város-sorszám (X, Y) és két napsorszám (P,Q) van egy-egy szóközzel elválasztva, ami azt jelenti, hogy az X. városból az Y városba az év P. és Q. napja között (P-t és Q-t beleértve) indulnak karaván. Az állomány utolsó sorában az indulási (A) és az érkezési (B) hely sorszáma van, valamint annak a napnak az éven belüli H sorszáma, amikor legkésőbb meg kell érkezni a B. városba, egy-egy szóközzel elválasztva. A karaván mindig reggel indulnak és még aznap este megérkeznek a célállomásra.

A karavan.ki állományba három sort kell írni! Minden sorban egyetlen szám szerepeljen: a részfeladat megoldásának értéke! Ha egy részfeladatnak nem létezik megoldása, akkor a –1 számot kell kiírni!

Példa:

karavan.be	karavan.ki	
5 7	3	(2. nap: 1→5, 3. nap: 5→4)
1 2 2 7	2	
1 5 2 4	7	(7. nap: 1→2, 8. nap: 2→5, 9. nap: 5→4)
2 3 1 2		
2 5 8 9		
3 4 6 7		
5 4 3 9		
4 2 1 2		
1 4 10		

3. feladat: Dominó (50 pont)

Dominóval sokféle játékot lehet játszani. Mohó Marci kedvenc dominós játéka a következő. Először véletlenszerűen sorba rakja a felhasználható dominókat. A játék célja az, hogy a lehető leghosszabb illeszkedő sorozatot képezzen a felhasználható dominókból. A játékszabály szerint minden lépésben csak a felhasználható dominósor első (bal oldali) elemét veheti és vagy elveti (félrerakja, de később nem veheti), vagy a már képzett illeszkedő sorozat bal vagy jobb végéhez teszi, feltéve, hogy az adott oldalával illeszkedik (megegyezik a pöttyök száma a két dominó érintkező oldalán). Az aktuális dominót mindkét oldalával próbálhatja illeszteni. A játék úgy kezdődik, hogy az első dominót ki kell raknia.

Készíts programot (Projekt név: domino (domino.cbp; main.cpp)), amely meghatározza a kirakható leghosszabb illeszkedő dominósor hosszát!

A domino.be állomány első sorában a felhasználható dominók száma (1≤N≤ 100000) van. A következő N sor mindegyikében egy dominó leírása, azaz két szám, X Y (0≤X,Y≤9) van egy szóközzel elválasztva. Bármely dominó (számpár) többször is szerepelhet az állományban, és az állomány nem feltétlenül tartalmaz minden lehetséges dominót.

A domino.ki állományba egyetlen számot kell írni, a kirakható leghosszabb illeszkedő dominósor hosszát!

Példa:

```
domino.be          domino.ki
6                  5
1 2
1 6
2 3
1 4
2 3
4 3
```

4. feladat: Alkimisták (50 pont)

Az alkimisták hosszas kísérleteiket arról, hogy milyen anyagok milyenekké alakíthatók át, gondosan feljegyezték könyveikbe. Bejegyzéseik a lehető legegyszerűbbek voltak: Megadták a kiindulási anyag nevét, majd azt az összetevőt (ún. katalizátort), amit hozzákeverve ehhez létrejött valamilyen végtermék. A bejegyzéseket tanulmányozva megállapították, hogy egyetlen anyag sem állítható elő önmagából, egy vagy több lépésben sem, továbbá nincs két különböző bejegyzés, amelyekben mind az első, mind a második anyag megegyezne. Az anyagok neveit számokkal, míg a katalizátorokat az angol ABC nagy betűivel jelölték A-tól Z-ig.

Egy ilyen bejegyzés sorozatra példa:

```
1 A 2     (1 anyagból 2 és
1 A 3      3 keletkezik, ha A-t keverünk hozzá)
1 B 4     (Ha 1-hez B-t adagolunk, akkor 4 keletkezik)
3 F 0     (Ha a létrejött 3-hoz F-et keverünk, akkor létrejön 0)
```

Az alkimisták aranyat szerettek volna előállítani vasból. Feljegyzéseikben az aranyat 0-val, míg a vasat 1-el jelölték.

Készíts programot (Projekt név: arany (arany.cbp; main.cpp)), amely megadja, hogy egy adott bejegyzés sorozatban melyek azok a katalizátorok, amelyek feltétlenül szükségesek ahhoz, hogy az alkimista szerint aranyat vasból elő lehessen állítani!

Az arany.be állomány első sorában megadjuk, hogy hány bejegyzés szerepel a sorozatban, a többi sora pedig a fenti példában szereplő formátumban tartalmazza az egyes bejegyzéseket (a kiindulási anyag száma, szóköz, a katalizátor betűjele, szóköz és a végtermék száma). A használt anyagok száma maximum 200, a katalizátoroké pedig 26. A bejegyzések száma legfeljebb 1000.

Az arany.ki állományba a nélkülözhetetlen katalizátorok betűjelét kell írni, egy sorban szóközzel elválasztva! Ha aranyat nem lehet előállítani semmilyen katalizátorral, akkor NEM LEHET szerepeljen a kimenetben! Ha egyik katalizátor sem nélkülözhetetlen, akkor EGYIK SEM KELL legyen a sor tartalma!

Példa:

```
arany.be           arany.ki
9                  A F G
1 A 2
1 A 3
1 B 4
3 F 5
5 G 6
6 A 7
6 B 8
7 C 0
8 D 0
```