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: OKTV (20 pont)

Seholsincs ország informatika OKTV-jén addig rendeznek új fordulókat, amíg vannak résztvevők, akiknek helyezése nem egyértelmű (holtversenyben vannak). Az újabb fordulókban csak ezeknek a résztvevőknek kötelező részt venniük, de új pontszámuk csak a holtverseny eldöntésére használható. (Az utolsó forduló után biztosan nincs holtverseny.) Részt vehetnek olyanok is a további fordulókban, akik helyezése már eldőlt. Ez utóbbi résztvevők ugyanúgy kapnak pontot, mint a többiek, de már nem változtathatnak helyezésükön. Ha úgy döntenek, hogy nem vesznek részt az újabb fordulóban, 0 pontot kapnak rá.

Az „oktv.be” állomány első sorában a versenyzők ($1 \le N \le 1000$) és a fordulók ($1 \le M \le 10$) száma van, a következő M sorban pedig az egyes fordulók eredménye. Minden eredménysor pontosan N nemnegatív számot tartalmaz, egy-egy szóközzel elválasztva.

Készíts programot (Projekt név: oktv (oktv.cbp; main.cpp)), amely az „oktv.ki” állományba és a képernyőre 1 sort ír, a versenyzők sorszámait helyezés szerinti sorrendben, egy-egy szóközzel elválasztva.

Példa:

```
oktv.be:
      5 3
      7 3 3 9 7
      2 7 7 0 1
      0 3 2 0 8
```

```
oktv.ki:
       4 1 5 2 3
```

2. feladat: Család (30 pont)

Családi kapcsolatokat úgy adunk meg, hogy mindenkihez felsoroljuk az anyja, illetve az apja nevét.

A „csalad.be” állomány első sorában az ismert személyek száma ($1 \le N \le 100$) van. Ezután 3*N sorban jönnek az egyes emberek adatai: a neve, az apja neve és az anyja neve. Az utolsó sorban egyetlen ismert név szerepel, akinek valamilyen rokonaira kíváncsiak vagyunk. A nevek hossza legfeljebb 20 karakter.

Készíts programot (Projekt név: csalad (csalad.cbp; main.cpp)), amely a képernyőre és a „csalad.ki” állományba az alábbi 4 sort írja:

A. A keresett személy testvérei száma és neve, egy-egy szóközzel elválasztva.

B. A keresett személy féltestvérei száma és neve, egy-egy szóközzel elválasztva. (A testvérek nem féltestvérek!)

C. A keresett személy férfiági felmenőinek listája (darabszám, majd apja, nagyapja, dédapja, ükapja, ... amíg ismert), egy-egy szóközzel elválasztva.

D. A keresett személy első unokatestvérei száma és neve (akikkel közös nagyszülője van az ismert személyek között), egy-egy szóközzel elválasztva.

Példa:

```
csalad.be:
7
Nagy Andrea
Nagy István
Kiss Anna
Nagy András
Nagy István
Kiss Anna
Nagy Erika
Nagy István
Szabó Éva
Nagy István
Nagy Péter
Kovács Eleonóra
Kiss Anna
Kiss Csaba
Takács Orsolya
Tóth Szilvia
Tóth Árpád
Takács Orsolya
Kovács Melinda
Kovács László
Tóth Szilvia
Nagy Andrea
```

```
csalad.ki:
1 Nagy András
1 Nagy Erika
2 Nagy István Nagy Péter
1 Kovács Melinda
```

3. feladat: Szójáték (25 pont)

A *Cserebere* logikai szójátékot egy szótár alapján játszhatjuk. A játék alapja: egy kiinduló szóból egy szósorozatot kell előállítani, amely csak a szótárban levő szavakból állhat, s a sorozat egymást követő szavai egy elemi átalakítással kaphatók az őket megelőző szóból. Kétféle szabály alkalmazható átalakításra:

A. szabály: egy betű kicserélése egy másikra;

B. szabály: egy betű kicserélése egy másikra vagy egy betű írása a szó végére.

A „jatek.be” állomány első sorában a szótár szavainak száma ($1 \le N \le 1000$), a további N sorban pedig egy-egy, a szótárban szereplő szó van. A szavak legfeljebb 20 karakterből állnak. A következő sorban az alkalmazandó szabály betűjele (A vagy B), az azt követő két sorban pedig két különböző, a szótárban szereplő szó (P és Q) van.

Készíts programot (Projekt név: jatek (jatek.cbp; main.cpp)), amely a képernyőre és a „jatek.ki” állományba írja soronként az alábbiakat:

1. Az első sorba az IGEN szó kerüljön, ha a P szóból előállítható a Q szó a megadott szabály szerinti szósorozat előállítással, egyébként a NEM szót kell írni. Ha előállítható, akkor az IGEN szótól egy szóközzel elválasztva szerepeljen a legkisebb szabályalkalmazás szám, amivel a Q a P-ből előállítható.

2. A második sorba a P szóból a szabály szerint (nem feltétlenül egyetlen sorozatban) előállítható összes szót kell írni.

Példa:

```
jatek.be:
7
OKOS
ÁKOS
ÁKOM
OKOD
ÁLOM
ALOM
HALOM
A
OKOS
ÁLOM
```

```
jatek.ki:
IGEN 3
OKOS ÁKOS ÁKOM ÁLOM ALOM OKOD
```

4. feladat: Kockák (25 pont)

Építőkockákból úgy lehet stabil tornyot építeni, hogy kisebb kockára nem lehet nagyobbat, illetve könnyebb kockára nem lehet nehezebbet tenni.

Készíts programot (Projekt név: kocka (kocka.cbp; main.cpp)), amely N kocka alapján megadja a belőlük építhető legmagasabb tornyot!

A „kocka.be” állomány első sorában a kockák ($1 \le N \le 1000$) száma van, a további N sorban pedig az egyes kockák oldalhossza és súlya (mindkettő 20000-nél kisebb pozitív egész szám), egyetlen szóközzel elválasztva.

A „kocka.ki” állomány és a képernyő első sorába a legmagasabb torony M kockaszámát kell írni, a következő M sorba pedig az építés szerint alulról felfelé sorrendben a felhasznált kockák oldalhosszát és súlyát.

Példa:

```
kocka.be:
5
10 3
20 5
15 6
15 1
10 2
```

```
kocka.ki:
3
20 5
10 3
10 2
```