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: Gombaszedés (32 pont)

Józsi bácsi hétvégénként nagy szeretettel jár a közeli erdőbe gombászni. Valamelyik hátizsákjával szokott elindulni, és ebbe a zsákba próbál meg minél több gombát szedni. Az erdőben háromféle gomba terem meg: Csiperke, Rókagomba és Lila pereszke. Józsi bácsi az évek során a következő módszert alakította ki a gombák leszedésére:

A. Mindaddig nem szed Lila pereszkét, amíg még van az erdőben Csiperke gomba.

B. Először mindig a legnagyobb (legnehezebb) gombát szedi le (persze, ha ez nem ütközik az A. ponttal).

C. Azonos súlyú gombák esetén először Csiperké(ke)t, majd a Rókagombá(ka)t, és legvégül a Lila pereszkéket szedi le.

Ha adott az erdő gombaállománya (minden gomba súlya dekagrammban és fajtája), és Józsi bácsi hátizsákjának kapacitása (azaz hány dekagramm gombát képes a zsákban tárolni) határozzuk meg, hogy hány dekagramm gombát tud (és ezekből fajtánként mennyit) leszedni. Tudjuk azt is, hogy Józsi bácsi nem vág ketté gombát, azért, hogy a zsákja még jobban tele legyen (tehát csak egész gombák vannak a zsákban). Ha egy nagyobb gomba már nem fér bele a zsákba, azt kihagyja.

Készíts programot (Projekt név: gomba (gomba.cbp; main.cpp)) a gombák zsákba pakolására!

A **`gomba.be`** állomány első sorában a gombák száma van (maximum 1000), a másodikban pedig a zsák kapacitása (maximum 100 000), a többi sor mindegyikében egy gomba jellemzői vannak: a fajta (a következő karakterek valamelyike: C, R vagy L) és a súly dekagrammban (1 és 100 között) egy szóközzel elválasztva.

A **`gomba.ki`** állományba a leszedett gombák darabszámát és súlyát kell írni (egy szóközzel elválasztva). Az első sorba az összes leszedett gomba, a másodikban csak a csiperke, a harmadikban a rókagomba, a negyedikben pedig a lila pereszke darabszámát és a súlyát.

Példa:

```
gomba.be
6
25
C 4
R 8
L 12
C 10
L 2
R 2
```

```
gomba.ki
4 24
2 14
2 10
0 0
```

Józsi bá' a következő sorrendben szedné le a gombákat: C 10, R 8, C 4, L 12, R 2, L 2. De sajnos a zsákja pici, ezért csak a C 10, R 8, C 4, R 2 gombák férnek el benne (az L

12 gombát kihagyja, mert a zsákba nem fér bele, így megpróbálja a sorban következőt belerakni).

2. feladat: Hálózat (24 pont)

Egy számítógépes hálózat kiépítése a következőképpen történik: Kezdetben egy gépből áll a hálózat. Egy új gép bekapcsolásakor azt pontosan 1 db, már a hálózatban levő géppel kötik össze, ami kétirányú kapcsolatot biztosít a két összekötött gép között.

Két gép távolságán a legkevesebb közvetlen összekötést tartalmazó összekötést értjük. A hálózat átmérőjének nevezzük a hálózatban levő gépek közül a két legtávolabbi távolságát.

Készíts programot (Projekt név: halozat (halozat.cbp; main.cpp)), ami kiszámítja, hogy N db gép esetén mekkora lesz a hálózat átmérője!

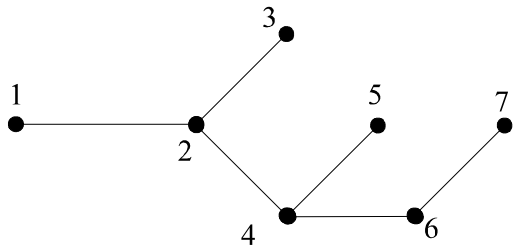
A **halozat.be** állomány első sorában N (1≤N≤100) értéke található. Az állomány I. sorában annak a számítógépnek a J (J<I) sorszáma van, amelyhez az I. számítógépet kötik a hálózatban.

A **halozat.ki** állományba egyetlen számot kell írni, a hálózat átmérőjét a teljes kiépülés után.

Példa:

```
halozat.be
7
1
2
2
4
4
6
```

```
halozat.ki
4
```



3. feladat: Zárnyitogató (18 pont)

Egy zár három körlemez tárcsából áll (A, B és C), amelyek közös tengely körül forgathatók. Az egyes tárcsákon körben az 1,..,N egész számok vannak felírva. A tárcsák külön-külön és együtt is forgathatók mindkét irányban. Tehát a zárat háromféleképpen fordíthatjuk el: egyszerre egy tárcsát (A, B vagy C), bármely kettő tárcsát egyszerre (egy irányban) (AB, vagy AC vagy BC), ill. a három tárcsát egyszerre (ABC) forgatva egy irányba.

Egy lépésnek a fenti forgatások valamelyikét nevezzük, tetszőleges irányba egy egység elfordulással.

Készíts programot (Projekt név: zar (zar.cbp; main.cpp)), amely a zár egy adott kiinduló helyzetéből ([a;b;c]) megadja a legrövidebb forgatási sorozat hosszát, amivel az [1;1;1] pozícióba juthatunk!

A **zar.be** állomány első sorában van az ***N*** (1≤***N***≤100) érték. A második sorban három szám van, egy-egy szóközzel elválasztva, a három tárcsán pillanatnyilag beállított szám.

A **zar.ki** állományba a legrövidebb lépéssorozat hosszát kell írni, amellyel a zár kinyitható, azaz az [1;1;1] pozícióba hozható.

Példa:

```
zar.be
6
1 5 3
```

```
zar.ki
4
```

Például egy lehetséges 4 hosszúságú lépéssorozat: BC-t kétszer forgatva csökkenő irányba, majd B-t kétszer forgatva csökkenő irányba.

4. feladat: Lift (26 pont)

Egy sokemeletes házban szokatlan módon üzemeltetik a liftet. A lift az első szintről indult és mindig felmegy a legfelső szintre, majd visszatér az első szintre. Menet közben megáll minden olyan szinten, amelyik uticélja valamelyik liftben tartózkodó utasnak. Hasonlóan, olyan szinten is megáll, ahonnan utazni szándékozik valaki az aktuális irányban, feltéve, hogy még befér a liftbe (figyelembe véve az adott szinten kiszállókat).

Készíts programot (Projekt név: lift (lift.cbp; main.cpp)), amely kiszámítja, hogy legkevesebb hány menet (1 menet = egyszer felmegy, majd lejön) szükséges ahhoz, hogy minden várakozó embert elszállítson a lift.

A **`lift.be`** bemeneti állomány első sorában két szám van, ***N*** és ***K***. ***N*** az épület szintjeinek ($2 \le N \le 100$) száma, ***K*** ($1 \le K \le 10$) pedig a lift kapacitása. A további ***N*** sor tartalmazza az egyes szinteken várakozó emberek adatait. Az állomány i-edik sorában azoknak a szinteknek a sorszáma van felsorolva, ahová az i-1–edik szintről utazni akarnak. A felsorolást minden sorban egy 0 szám zárja. Minden sorban legfeljebb 200 szám lehet, tetszőleges sorrendben.

A **`lift.ki`** állomány egyetlen sort tartalmazzon, a legkevesebb menetek számát, amely az emberek elszállításához szükséges!

Példa:

lift.be	lift.ki
6 2 2 3 2 0 1 3 0 1 2 0 2 5 0 3 6 2 0 1 2 3 0	3