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: Memória (25 pont)

A számítógép memóriájának szabad területeit egy N elemű sorozatként tartjuk nyilván, memóriacím szerint növekvő sorrendben (N a szabad területek száma). Minden sorozatelem egy kettős: kezdőcím, hossz. Mindkettő 1 és 100 millió közötti egész szám.

Memória-felszabadítás esetén újabb szabad területek keletkezhetnek, amelyeket be kell venni ebbe a sorozatba a kezdőcím szerinti helyére, kivéve, ha összeérne valamelyik korábbi szabad területtel (ekkor össze kell olvasztani őket egyetlen, nagyobb szabad területté). Memória-lefoglalás esetén az első olyan szabad területből adunk memóriát, amiben legalább az igénynek megfelelő memória van. Feltehető, hogy ilyen mindig van!

Írj programot, amely elvégzi a kért műveleteket!

A memória aktuális állapota és az igények megtalálhatók a `memoria.be` állományban. Első sora az N értékét ($1 \le N \le 10\,000$), következő N sora a sorozatelemeket tartalmazza egy szóközzel elválasztva. A következő sorban a műveletek M száma található ($1 \le M \le 1000$). Az ezt követő rész soronként egy betűvel kezdődik: F jelenti a felszabadítást, L jelenti a lefoglalást. F esetén a betűt egy szóközzel elválasztott számpár követi: a felszabadítandó terület kezdőcíme és hossza. L esetén egyetlen szám szerepel a sorban: a lefoglalandó terület hossza.

A memória aktuális állapotát leíró sorozatot a `memoria.ki` állományba kell írni minden művelet után, a bemenetnek megfelelő formátumban!

Példa:

`memoria.be`

```
3
1000 1000
3000 500
5000 2500
4
F 2500 100
F 4000 1000
L 100
L 900
```

`memoria.ki`

```
4
1000 1000
2500 100
3000 500
5000 2500
4
1000 1000
2500 100
3000 500
4000 3500
4
1100 900
2500 100
```

```
3000 500
4000 3500
3
2500 100
3000 500
4000 3500
```

2. feladat: Város (25 pont)

Egy modern nagyváros úthálózata egy négyzetráccsal írható le, ahol N jelöli a négyzetrács sorainak számát (azaz a kelet-nyugati irányú utak számát, az ilyen utakat balról jobbra sorszámozzuk), M pedig az oszlopokét (azaz az észak-déli utakét, az ilyen utakat alulról felfelé sorszámozzuk). El szeretnénk jutni a város egyik kereszteződéséből egy másik kereszteződésbe. Az egyes kereszteződések (csomópontok) előtt a haladási irányt befolyásoló jelzőtáblák lehetnek, melyeket a következő kódokkal látunk el:

balra fordulni tilos	BT
jobbra fordulni tilos	JT
balra haladni kötelező	BK
jobbra haladni kötelező	JK

Írj programot, amely a táblák figyelembevételével megadja a legrövidebb útvonalat, amelyeken áthaladva eljuthatunk az indulási helyről a célba! Útközben a várost nem hagyhatjuk el (bár erről szóló jelzőtáblák nincsenek) és vissza sem lehet fordulni az adott négyzetrácsból ugyanoda.

A `varos.be` állomány első sorában a sorok és oszlopok száma ($1 \le N,M \le 100$), valamint a táblák száma ($1 \le T \le 100\,000$) van egy-egy szóközzel elválasztva. A következő T sorban soronként egy-egy tábla leírása található. A táblaleírás formája: `kód sor oszlop irány`, ahol a `sor` és az `oszlop` a csomópont koordinátáit adja meg ($1 \le sor \le N$, $1 \le oszlop \le M$), az `irány` pedig a csomópontba beérkező útszakasz irányát (`E`, `K`, `D`, `N`). Az utolsó sorban a két kereszteződés sor és oszlopindexe van, egy-egy szóközzel elválasztva, az első az induló hely, a második a cél.

A `varos.ki` állomány első sorába a két pont közötti legrövidebb út hosszát kell írni! A második sorba a legrövidebb út leírását kell írni, ahol minden lépést a haladás iránya, azaz az E, K, D vagy N betű azonosít.

Példa:

`varos.be`

```
100 100 5
JK 2 1 E
JK 2 2 K
BT 2 2 E
BT 4 2 E
JT 4 2 E
1 1 4 1
```

`varos.ki`

```
5
KEENE
```

3. feladat: Mingrel (25 pont)

A mingrel ábécé betűi szó szerint megegyeznek az angol ábécé betűivel, a betűk ábécé-sorrendje azonban eltér az angolétól.

Készíts programot, amely mingrel szavak ábécé sorrendje alapján megadja a mingrel ábécé betűinek lehetséges ábécé-sorrendjét!

A `mingrel.be` állomány első sora az ismert mingrel szavak számát tartalmazza ($1 \le N \le 10\,000$), majd soronként egy-egy szót a mingrel ábécé szerinti sorrendben, csupa kisbetűvel írva.

A `mingrel.ki` állomány egyetlen sorába a mingrel ábécé betűinek egy lehetséges ábécé-sorrendjét kell írni!

Példa:

`mingrel.be`

```
5
atom
alma
utt
utas
ember
```

`mingrel.ki`

```
tauelombcdfghijknpqrsvwxyz
```

4. feladat: Kulcsszavak (25 pont)

Egy könyvtár a kulcsszavakat tárol, amelyek megkönnyítik a könyvek keresését.

Készíts programot, amely egy kulcsszó-kifejezés alapján megadja azon könyvek sorszámát (1-től kezdődően), amelyek a kifejezésnek megfelelnek.

A kulcsszavak a `kulcs.be` állományban találhatók. Az állomány első sorában a könyvek N száma van ($1 \le N \le 1000$). A következő N sorban az egyes könyvek kulcsszavai szerepelnek szóközökkel elválasztva, egy könyvhöz legalább 1 és legfeljebb 10. Az állomány utolsó sorában egy kulcsszó-kifejezés van.

A kulcsszó-kifejezés vagy egyetlen kulcsszó; vagy &-jellel (ÉS kapcsolat) összekapcsolt kulcsszavak (pl. `alma&körte&barack`), melyek mindegyikének szerepelnie kell az adott könyv kulcsszavai között; vagy az előbbiek + jellel (VAGY kapcsolat) összekapcsolva (pl. `fa&bokor+fű`), és a + jelekkel összekapcsolt kulcsszó-csoportok közül valamelyiknek kell szerepelnie az adott könyv kulcsszavai között.

A `kulcs.ki` állomány első sorába azon könyvek számát kell írni, amelyek megfelelnek a kulcsszó-kifejezésnek! A második sorba pedig ezen kifejezésnek megfelelő könyvek sorszáma kerüljön, növekvő sorrendben! Ha nincs megoldás, mindkét sorba 0 kerüljön.

Példa:

`kulcs.be`

```
5
zebra kutya
kutya tigris farkas
ponty csuka
zebra antilop kecske
farkas medve
zebra+kutya&farkas
```

`kulcs.ki`

```
3
1 2 4
```