**Проблем 1. Тројке**

Професор математике је поставио Драганчету следећи задатак. На основу датог низа бројева $a_1, a_2, ..., a_n$, Драганче треба да за сваку тројку индекса $(i, j, k)$, где је $1 \leqslant i < j < k \leqslant n$, напише на табли највећи од бројева $a_i, a_j$ и $a_k$. Затим треба да израчуна остатак који даје збир свих бројева који су написани на табли при дељењу са 10007. Професор је обећао Драганчету петицу за крај школске године, ако добије тачно решење пре краја часа. Помозите Драганчету да што брже добије тачан резултат.

**Улаз.** (Улазни подаци се налазе у датотеци `trojke.in`) У првом реду улазне датотеке налази се број $n$ ($3 \leqslant n \leqslant 30000$). У следећих $n$ редова се налазе цели бројеви $a_1, a_2, ..., a_n$, при чему је $-100000 \leqslant a_i \leqslant 100000$.

**Излаз.** (Излазне податке уписати у датотеку `trojke.out`) У првом и једином реду излазне датотеке исписати суму бројева написаних на табли по модулу 10007.

**Пример 1.**

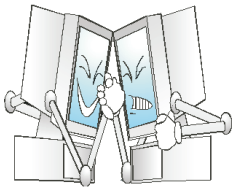
| trojke.in | trojke.out |
|---|---|
| 4<br>3<br>-1<br>2<br>2 | 11 |

**Објашњење.** Све тројке низа бројева су: (3, -1, 2), (3, -1, 2), (3, 2, 2), (-1, 2, 2). На табли су написани бројеви 3, 3, 3, 2, па је решење у овом случају 11.

**Пример 2.**

| trojke.in | trojke.out |
|---|---|
| 6<br>8<br>-10<br>4<br>5<br>2<br>6 | 135 |

**Ограничења.** Максимално време извршавања програма је 1 секунда.





**Проблем 2. Порука**

Мали Ђурица куца поруку на мобилном телефону. Он је откуцао $n$ редова. Када мали Ђурица каже да се курсор налази на позицији $(i, j)$, то значи да се налази у $i$-том реду иза $j$-тог знака - ако је $j$ нула, то значи да је курсор на почетку $i$-тог реда. Малог Ђурицу занима колико најмање пута треба да притисне тастере горе, доле, лево, десно (који служе за померање курсора кроз поруку) тако да од позиције $(sr, sc)$ стигне до позиције $(fr, fc)$. Курсор сем позиције садржи и вредност последње позиције у реду која је добијена притиском тастера лево или десно, чије ће значење детаљно бити описано у наставку проблема. Ову вредност ћемо звати poslPoz.

Мали Ђурица ће Вам рећи колико редова има у његовој поруци и колико је дугачак сваки ред, тј. $len_i$ број који представља број знакова у $i$-том реду (односно дужину $i$-тог реда). Он Вам такође шаље на који начин функционишу његови тастери и његов мобилни:

1. Следећи ред последњег реда је први ред; претходни ред првог реда је последњи ред, односно редови су организовани циклично. Преласком у следећи, односно претходни ред, курсор ће се наћи на позицији коју показује poslPoz у складу са правилом 8.

2. Притиском на тастер доле курсор прелази у следећи ред; притиском на тастер горе курсор прелази у претходни ред.

3. Притиском на тастер горе или доле вредност poslPoz се не мења.

4. Ако се курсор налази на почетку реда $r$, притиском на тастер лево курсор прелази на позицију иза последњег знака претходног реда и вредност poslPoz се мења на вредност позиције иза тог знака.

5. Ако се курсор не налази на почетку реда $r$, тада притиском на тастер лево за позиције $(r, c)$, прелази на $(r, c-1)$, а poslPoz постаје $c-1$.

6. Ако се курсор налази иза последњег знака реда $r$, притиском на тастер десно курсор прелази на почетак следећег реда и вредност poslPoz постаје 0.

7. Ако се курсор не налази иза последњег знака реда $r$, тада притиском на тастер десно са позиције $(r, c)$, прелази на $(r, c+1)$, а poslPoz постаје $c+1$.

8. Ако се у неком моменту курсор нађе у реду који има $c$ карактера, а вредност poslPoz је већа од $c$, тада ће курсор бити на позицији $c$ у том реду. Вредност poslPoz СЕ НЕ МЕЊА у том случају.

**Ограничења.** Број редова није већи од 100. Дужина редова није већа од 100.

**Улаз.** (Улазни подаци се налазе у датотеци `poruka.in`) У првом реду се налази природан број $n$ ($1 \leqslant n \leqslant 100$). Потом се у $n$ редова налазе цели бројеви $len_i$ ($0 \leqslant len_i \leqslant 100$) који редом означавају дужине редова. Затим се у $n+2$-ом реду налазе природни бројеви $sr$ ($1 \leqslant sr \leqslant n$), $sc$ ($0 \leqslant sc \leqslant len_{sr}$), $fr$ ($1 \leqslant fr \leqslant n$) и $fc$ ($0 \leqslant fc \leqslant len_{fr}$). Прва два од тих поља означавају полазну позицију (редни број врсте и колоне), а друга два завршну позицију.

**Излаз.** (Излазне податке уписати у датотеку `poruka.out`) У излазну датотеку исписати минимални број притисака тастера да се од позиције $(sr, sc)$ дође до позиције $(fr, fc)$ користећи наведена правила.

**Пример 1.**

| poruka.in | poruka.out |
|---|---|
| 5<br>1<br>3<br>1<br>3<br>1<br>4 3 2 3 | 2 |

Порука би могла да изгледа овако

```
X
XXX
X
XXX
X
```

а курсор се налази иза болдованог слова. Позиција је (4, 3), poslPoz има вредност 3. Тада притиском горе курсор прелази на крај 3. реда и има позицију (3, 1), а poslPoz остаје иста. Поново притиском горе прелази у 2. ред, а курсор прелази на позицију у реду коју показује poslPoz и курсор се налази на жељеној позицији.

**Пример 2.**

| poruka.in | poruka.out |
|---|---|
| 7<br>10<br>100<br>100<br>100<br>100<br>100<br>100<br>2 50 3 10 | 5 |

**Објашњење.** Порука би могла да изгледа овако

```
XXXXXXXXXX
XX...XX - укупно 100 знакова X
XX...XX - укупно 100 знакова X
... - 6 редова, сваки са 100 знакова X
XX...XX - укупно 100 знакова X.
```

Притиском на тастер горе курсор прелази на (1, 10); poslPoz је 50. Притиском на тастер лево курсор прелази на (1, 9); poslPoz постаје 9. Притиском на тастер десно курсор прелази на позицију (1, 10); poslPoz постаје 10. Притиском два пута на тастер доле курсор прелази на позицију (2, 10), па (3, 10).

**Ограничења.** Максимално време извршавања програма је 0.2 секунде.

**Проблем 3. Најмањи**

Дат је низ бројева $a_1, a_2, ..., a_n$. Под конкатенацијом два броја $x$ и $y$ подразумевамо број $xy$, који је добијен надовезивањем цифара броја $y$ после броја $x$ (нпр. конкатенацијом бројева 123 и 45 добијамо број 12345). Велики број добијамо када дописујемо бројеве један иза другог у неком редоследу. Одредити најмањи број који се добија конкатенацијом свих бројева $a_1, a_2, ..., a_n$.

**Улаз.** (Улазни подаци се налазе у датотеци `najmanji.in`) У првом реду улазне датотеке налази се цео број $n$ ($2 \leqslant n \leqslant 5000$). У следећих $n$ редова се налазе природни бројеви $a_1, a_2, ..., a_n$ ($1 \leqslant a_i \leqslant 2000000000$). Бројеви су задати без почетних (неважећих) нула.

**Излаз.** (Излазне податке уписати у датотеку `najmanji.out`) У првом реду излазне датотеке исписати најмањи број који се добија конкатенацијом учитаних бројева.

**Пример 1.**

| najmanji.in | najmanji.out |
|---|---|
| 2<br>91919<br>919191 | 91919191919 |

**Пример 2.**

| najmanji.in | najmanji.out |
|---|---|
| 5<br>32<br>11<br>987<br>12<br>3 | 1112323987 |

**Објашњење.** Најмањи број добијамо конкатенацијом редом 11, 12, 32, 3, 987.

**Ограничења.** Максимално време извршавања програма је 1 секунда.

**Проблем 4. Лет**

Испред Вас се налази екран издељен на $k$ ветикалних трака које су с лева на десно редом означене бројевима $1, 2, ..., k$. На дну екрана се налази летелица која може да се креће кроз траке (из једне траке летелица може да пређе само у суседну), а којој треба 1 секунда да се помери у следећу траку екрана. Са горњег дела екрана пада $n$ дијаманата. За сваки дијамант је позната цена $c$, трака $l$ у којој пада (дијамант пада у тачно једној траци) и моменат $t$ у којем ће дотаћи дно екрана. Летелица је ширине траке и може да покупи неки дијамант само ако се целом ширином налази у траци у којој пада дијамант тачно у времену када дијамант дотакне дно екрана. Ако летелица у моменту $x$ крене да прелази у суседну траку, тада ће се целом својом ширином у суседној траци наћи у моменту $x + 1$. Значи, ако у моменту $x$ са траке $s$ крене да прелази у суседну траку $s1$ ($s1$ је или $s - 1$ или $s + 1$), тада ће у траци $s$ покупити дијаманте који у тренутку $x$, а у $s1$ дијаманте који у тренутку $x + 1$ дотичу дно екрана. Док се летелица налази у једној траци целом ширином, у тој траци може да остане произвољно време.

Летелица може да скупља дијаманте $t$ секунди. У моменту 0 летелица се налази у траци 1. Ваш задатак је да нађете максималну цену дијаманата коју летелица може да покупи за дато време.

**Улаз.** (Улазни подаци се налазе у датотеци `let.in`) У првом реду се налазе природни бројеви бројеви $k$ ($1 \leqslant k \leqslant 50$), $n$ ($1 \leqslant n \leqslant 100000$) и $T$ ($1 \leqslant T \leqslant 100000$). У наредних $n$ редова се налази подаци о дијамантима. У $i + 1$-ом реду се налазе природни бројеви $c_i$ ($1 \leqslant c_i \leqslant 1000000$), $l_i$ ($1 \leqslant l_i \leqslant k$) и $t_i$ ($1 \leqslant t_i \leqslant 200000$), који редом означавају цену дијаманта, трака у којој дијамант пада и моменат у којем ће дијамант дотаћи дно екрана.

**Излаз.** (Излазне податке уписати у датотеку `let.out`) У првом реду исписати максималну цену дијаманата које летелица може да покупи.

**Пример 1.**

`let.in`

```
5 11 10
10 1 2
10 1 2
200 3 2
50 3 2
50 3 2
10 4 2
10 4 2
200 5 5
50 1 4
10 2 2
10 2 2
```

`let.out`

```
500
```

**Пример 2.**

let.in
```
4 9 10
200 4 1
200 4 3
5 1 1
5 1 2
5 1 3
5 1 3
5 1 4
5 1 5
5 1 11
```

let.out
```
200
```

**Објашњење.** На почетку игре летелица почиње кретање према траци 4. У траку 4 стиже у 3. секунди и купи дијамант цене 200. Након тога нема времена да покупи још нешто (приметимо да се игра завршава после 10 секунди, па дијамант који ће на дно екрана доспети у 11. секунди не може бити ухваћен).

**Ограничења.** Максимално време извршавања програма је 1.5 секунда.

**Проблем 5. Огрлице**

Наш мали Драганче се нашао у невољи. Пре пар недеља се договорио са другарима да на лето иде на море у Абенишбе, али је убрзо схватио да нема довољно пара. Па је одлучио да се запосли и да заради те паре. Пошто га нико није схватио озбиљно да са 10 година уме да програмира, морао је да се запосли на неком много мање занимљивом месту - у продавници накита. У тој продавници имају јако велики избор - огрлице, минђуше, наруквице, прстење... Нашем малом Драганчету за око су посебно запале огрлице од перли. Перле су поређане у круг, и са неких перли из круга виси још по једна ниска перли. Сваке две суседне перле су повезане малим кончићем (и са круга и са ниски). Драганче је приметио да је огрлица јако уска, тако да ретко која муштерија може да је стави око врата, тако да и они којима се свиди, одустану после пробавања. Драганче је смислио како да реши проблем! Маказама ће пресећи један кончић (који спаја две перле са круга), и неке две перле ће да споји кончићем. Тако ће да добије огрлицу истог типа - круг са висећим нискама перли. Пошто већ пар недеља није ништа програмирао, јер сваки дан седи у продавници, замолио вас је да му помогнете, и израчунате колики највећи круг на огрлици може да добије, са једним сечењем и једним спајањем. Наравно, када би знао највећи круг, могао би свакој огрлици да повећа круг, и самим тим повећа број муштерија.

**Улаз.** (Улазни подаци се налазе у датотеци `ogrlica.in`) У првом реду улазне датотеке се налази број $n$ ($n \leqslant 500000$) који представља број перли на кругу. У следећем реду су два броја, $k$ и $m$ ($k \leqslant 10000$, $m \leqslant 10000000$). У следећих $k$ редова се налази по један цео број низа $x_i$ ($x_i \leqslant 2 \times m$). Низ $a_i$ израчунајте користећи доеле наведени сегмент програма (низ $x_i$ има $k$ елемената и њихови индекси су од 0 до $k-1$, а $a_i$ има $n$ елемената и њихови индекси су од 0 до $n-1$):

```
j = 0;
for (i = 0; i < n; i++) {
   a[i] = x[j];
   s = (j+1) % k;
   x[j] = ((x[j] ^ x[s]) + 13) % m;
   j = s;
}
```

(% означава остатак по модулу, а ^ означава битовски xor)

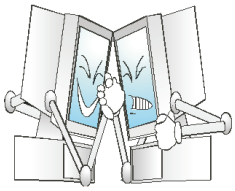
За `Pascal` програмере би сегмент имао следећи изглед:

```
j := 0;
for i := 0 to n-1 do begin
   a[i] := x[j];
   s := (j+1) mod k;
   x[j] := ((x[j] xor x[s]) + 13) mod m;
   j := s;
end;
```

У овом коду је xor ознака за битовну екслузивну дисјункцију која постоји као операција у Rascal-у.

Број $a_i$ представља број перли у ниски испод $i$-те перле на кругу. Решење не зависи од горње формуле, у њој не постоје зависности које би вам помогле у решавању. Она служи да улазна датотека не буде превелика, да би могла да се учита у временском ограничењу.

**Излаз.** (Излазне податке уписати у датотеку `ogrlica.out`) У први ред излазне датотеке исписати један цео број - број перли у највећем кругу који се може добити једним сечењем и једним спајањем две перле дате огрлице.

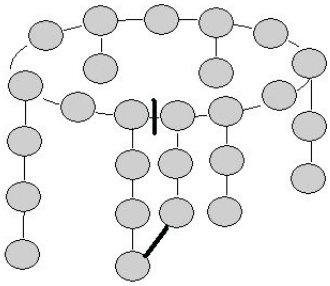




**Пример 1.**

ogrlice.in
```
12
12 5
3
0
3
2
2
0
2
0
1
0
1
0
```

ogrlica.out
```
17
```



**Објашњење.** Огрлица је приказана на слици испод одговора. Низови $x$ и $a$ су идентични. Ако пресечемо између 2. и 3. перле са круга, и спојимо последње перле из ниски испод 2. и 3. перле, добијамо дужину $12 + 3 + 2 = 17$.

**Пример 2.**

ogrlice.in
```
8
4 9
3
4
0
5
```

ogrlica.out
```
20
```

Низ a, који треба да се добије када генеришете је $3, 4, 0, 5, 2, 8, 0, 2$.

**Ограничења.** Максимално време извршавања програма је 0.2 секунде.