КЛАРКСОН

"Jeremy Clarkson Beatbox" е популарна YouTube монтажа од поранешниот водител на шоуто Top Gear кое го емитува BBC. Видеото содржи низа од сцени од шоуто монтирани една по друга, со цел да се добие забавен музички перформанс – битбокс. Ова видео беше објавено на YouTube во 2009 и оттогаш е прегледано повеќе од 3 милиони пати.

Оваа година Jeremy Clarkson го напушти BBC. За да се обележи неговото историско шоу, сакаме да направиме продолжение на популарната монтажа користејќи сцени од последните епизоди од шоуто. Текстот на новото видео е веќе познат – неговите фанови веќе го имаат избрано. Но, правењето на монтажата не е многу лесно (треба да се лепат, сечат и монтираат сцените).

Проблемот во монтирањето е тоа што без разлика колку умешно се монтирани сцените едно по друго, гледачите секогаш ќе можат да ги забележат транзициите во видеото (премините од една во друга сцена). Некои гледачи може да се иритираат од ова видео ако во краток период има две транзиции. Затоа, целта на монтажата е да се обезбеди транзициите да се што е можно поодделечени една од друга со тоа што ќе се максимизира должината на најкратката сцена во монтажата.

Задача

Напишете програма која има две линии на влез: во првата линија е текстот на видеото кое треба да се креира, а во втората линија е транскриптот (текстот) од последните епизоди на шоуто Top Gear. Вашата програма треба да го најде најдобриот начин на кој може да се добие текстот на видеото од подстринговите во транскриптот така што ќе се максимизира должината на најкраткиот подстринг. Ваквата монтажа велиме дека е оптимална. Ако монтажата е оптимална, излезот на програмата е должината на најкраткиот подстринг. Ако не може да се направи монтажа од дадениот транскрипт, излезот на програмата треба да е -1.

Влез

Од стандарден влез се читаат две линии текст (текстот на видеото и транскриптот од епизодите). Текстот се состои само од мали и големи букви од англиската азбука.

Излез

На излез треба да се отпечати еден цел број кој ја претставува должината на најкраткиот подстринг во оптималната монтажа на видеото. Ако не може да се монтира саканото видео излезот е -1.

Ограничувања

Во секоја линија влезот има најмногу 100 000 знаци.

Пример

Влез

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

Излез

```
3
```

Објаснување

```
Just|ice|AndTr|ust
Min(4, 3, 5, 3) = 3
CarsAndTrucksAreJustNice
```

Подзадачи

Нека *N* е должината на текстот во првата линија, а *M* должината на текстот во втората линија од влезот.

подзадача	поени	*Максимумот од N и M е помал или еднаков на*	Забелешка
1	11	**10**	
2	17	**2 000**	
3	19	**10 000**	
4	23	**30 000**	Одговорот (излезот) на проблемот е помал или еднаков на 20
5	30	**100 000**	

РАДИО

Граница помеѓу Бугарија и Романија е реката Дунав. Всушност, веројатно веќе ја видовте – многу тешко е да ја промашите.

Бидејќи е можно да се премине преку реката со чамец, луѓето во минатото го правеле и сеуште го прават тоа. За да се спречат сообраќајни несреќи, спасувачката служба изградила **N** безбедносни кули на бугарскиот брег. Позициите на овие кули се оддалечени $\mathbf{X_1}$, $\mathbf{X_2}$, …, $\mathbf{X_N}$ метри по течението на Дунав, од точката каде што реката влегува во Бугарија. Заради поедноставување, ќе сметаме дека Дунав претставува права линија, а кулите – точки на кои се придружени соодветни целобројни координати.

Во секоја кула има по еден радио предавател (анг. radio transmitter). Предавателот во кулата *i* има моќност $\mathbf{P_i}$. Користејќи ги овие предаватели кулите би можеле да комуницираат една со друга. Две кули *i* и *j* можат да комуницираат ако растојанието помеѓу нив е помало или еднакво на збирот на моќностите на нивните предаватели (со други зборови: ако $|\mathbf{X_i} - \mathbf{X_j}| \leq \mathbf{P_i} + \mathbf{P_j}$).

Со цел да се редуцираат трошоците, била донесена одлука да се отстранат некои од кулите. Новиот план вели дека треба да останат само **K** кули, а преостанатите ќе бидат продадени. Продажбата на кулата *i* и носи на владата $\mathbf{S_i}$ лева (лев, множина: лева, е бугарската парична единица), но продадената кула повеќе не може да се користи.

Како и да е, за да се добие еден добро функционален систем, воведено било уште едно ново побарување: секои две од преостанатите **K** кули мора да можат да комуницираат помеѓу себе директно (ако остане само една кула, ги занемаруваме комуникациите). Бидејќи ова може да е невозможно со нивните актуелни моќности, кулите може да се надградуваат. Надградбата на која било кула ја зголемува нејзината моќност по цена од еден лев за една единица моќност.

Задача

Борис неодамна започнал со летна пракса во Министерството за финансии. Сега тој сака да се истакне со тоа што ќе предложи најисплатлив план за продажба на **N** − **K** кули и надградба на преостанатите, така што секој пар од преостанатите кули ќе може директно да комуницира. Вие одлучувате да му помогнете на Борис да се истакне со

тоа што ќе напишете програма која ќе ја пронаоѓа минималната цена (или максималната добивка, доколку продажбата на кулите донесе повеќе пари во однос на парите што се потрошени за надградба) за оваа задача.

Влез

Во првата линија од стандардниот влез се дадени два цели броеви **N** и **K** – бројот на кули на почетокот и на крајот, соодветно. Во секоја од следните **N** линии дадени се три цели броеви $\mathbf{X_i}$, $\mathbf{P_i}$, $\mathbf{S_i}$ – позицијата на соодветната кула, нејзината почетна моќност, како и цената за која што може да се продаде таа. Кулите се дадени во растечки редослед според нивните позиции $\mathbf{X_i}$. Не постојат две кули кои што имаат идентични координати $\mathbf{X_i}$.

Излез

На единствената линија од стандардниот излез, вашата програма треба да отпечати точно еден цел број – минималната цена (или максималната добивка) за извршување на задачата. Во случај на добивка, отпечатениот број треба да биде негативен.

Ограничувања

- $1 \le \mathbf{K} \le \mathbf{N} \le 100\,000$
- $1 \le \mathbf{X_i}, \mathbf{P_i}, \mathbf{S_i} \le 1\,000\,000\,000$

Подзадачи

- Подзадача 1 (15 поени): $\mathbf{N} \le 15$;
- Подзадача 2 (15 поени): $\mathbf{N} \le 1\,000$, $\mathbf{N} = \mathbf{K}$;
- Подзадача 3 (15 поени): $\mathbf{N} \le 1\,000$, $\mathbf{S_i} = 1$;
- Подзадача 4 (15 поени): $\mathbf{N} \le 100\,000$, $\mathbf{N} = \mathbf{K}$;
- Подзадача 5 (15 поени): $\mathbf{N} \le 100\,000$, $\mathbf{S_i} = 1$;
- Подзадача 6 (25 поени): $\mathbf{N} \le 100\,000$.

Примери

Влез	Влез
5 3 4 63 3 13 2 4 87 3 9 121 6 15 159 5 2	9 5 5 8 4 10 10 7 11 9 7 13 6 6 19 20 9 20 2 1 23 1 3 26 13 11 28 4 2
Излез	**Излез**
42	-24

Објаснување

Во првиот пример, една оптимална варијанта е да останат кулите со индекси {1, 3, 4}. На ваков начин, ќе треба да се зголеми моќноста на кулата 1 за 17 единици и моќноста на кулата 4 за 31 единица, што ќе чини 17 + 31 = 48 лева. Продавајќи ги преостанатите кули (2 и 5), се добиваат (заработуваат) 4 + 2 = 6 лева. Вкупната „цена“ е 48 – 6 = 42 лева.

Во вториот пример, може, на пример, да се избере да останат кулите со индекси {2, 3, 6, 7, 9}. За да можат кулите да комуницираат една со друга, ќе мора да се зголеми моќноста на кулата 7 за 2 единици и моќноста на кулата 9 за 4 единици, по цена од 2 + 4 = 6. Со ова множество на кули може да се продадат {1, 4, 5, 8} и да се заработат 4 + 6 + 9 + 11 = 30 лева. Вкупната „цена“ е 6 – 30 = −24 лева. Според тоа, се добива заработувачка од 24 лева.

ПОПЛОЧУВАЊЕ

Нека k, l и n се позитивни цели броеви. Една правоаголна соба има под со димензии $k{\cdot}n \times l{\cdot}n$ cm (знакот $\cdot$ е знак за множење). Подот треба да се поплочи со $k{\cdot}l{\cdot}n$ правоаголни плочки со димензија $1 \times n$. При поплочувањето ниту една плочка не е пресечена (плочките се стават цели). Да забележиме дека подот е всушност правоаголна мрежа (анг. *grid*) со $k{\cdot}n$ колони и $l{\cdot}n$ редици.

Еден робот, кој сака да привлече внимание, се заклучил во собата и се заканува дека ќе ја уништи собата ако не погодите како точно е поплочен подот: т.е. која е точната положба на секоја плочка со која е поплочен подот. За среќа, роботот мора да ви даде одредена информација, врз база на која вие може да откриете како точно е поплочен подот. Вие можете да го прашате роботот q прашања од тип (x y), каде $1 \le x \le k{\cdot}n$ и $1 \le y \le l{\cdot}n$ се броеви со кој е означена една колона и редица, соодветно (броењето на редиците и колоните почнува од горниот лев агол на собата , па горниот лев агол е во прва колона и прва редица (1 1)).

За прашањата ќе добиете q ***точни*** одговори во кои ќе добиете точен опис за положбата на плочката која е залепена на секое поле од подот (x y) за кое е поставено соодветно прашање. Секој одговор ќе содржи: координати на горниот лев агол (“почетокот”) на плочката и начинот на која таа е залепена (“правецот”) – “хоризонтално” (во една редица) или “вертикално” (во една колона).

Задача

Напишете функција `tiling()`, која ќе се компајлира со програмата оценувач и која ќе води краток дијалог со роботот за да го реконструира поплочувањето од добиените информации (т.е. за да најде како точно е поплочен подот).

Една идеја е да го поставите прашањето за секое поле од подот и тогаш ќе го имате решението! Но, бројот на прашања q кои ќе му ги поставите на роботот треба да е најмалиот можен затоа што во спротивно ризикувате да го изнервирате роботот. Ако го изнервирате роботот, дури и во случај да погодите како точно е поплочен подот, роботот ќе ја уништи собата (во овој случај добивате 0 поени на тест случајот: погледнете ги правилата за оценување).

Имплементациски детали

Треба да ја предадете датотеката **tiling.cpp**, во која се наоѓа функцијата `tiling()`. Во оваа датотека може да имате и други потребни делови од код, но не смее да ги користите како глобални ниту имињата предефинирани подолу, ниту пак името `main`.

На почетокот на датотеката треба да стои претпроцесорската директива

```
#include "tiling.h"
```

Во програмата оценувач се декларирани и имплементирани следните елементи:

```
struct Data
{int x,y;
 char d;
};
void getArea(int *k, int *l, int *n);
bool getData(int q, Data *quest);
void setResult (const char *res);
```

Опис на дефинираните функции за комуникација со роботот:

1. Функцијата **`getArea`** ги доделува точните вредности за параметрите на собата *k*, *l* и *n* (за овие параметри бројот на колони ќе биде *k*·*n*, а бројот на редици *l*·*n*).
2. Функцијата **`getData`** може да се повика **САМО ЕДНАШ**. Параметрите го имаат следното значење:

- *q* е бројот на прашања;
- Влезно/излезната низа од записи `quest` ги содржи:
  - Пред повикот на функцијата – самите прашања (*x* и *y* се соодветно колоната и редицата на полето за кое се поставува прашањето). Да забележиме дека членката на записот *d* нема никакво значење тука.
  - По повикот на функцијата – одговорите на роботот. Имено, *x* и *y* се колоната и редицата, соодветно, на „почетокот“ на плочката која е залепена над полето за кое е поставено прашањето. Членката *d* како вредност има еден од следните два знака: `h` – ако плочката е залепена хоризонтално, или `v` – ако плочката е залепена вертикално. Ако функцијата врати вредност **`false`**, тоа значи дека или податоците од низата се неточни (на пример, вредностите се надвор од границите) или функцијата веќе била повикана. Во овој случај низата ќе содржи нули како излезни вредности.

3. Пред самиот крај на функцијата `tiling()`, треба да се повика дефинираната функција **`setResult`**, така што аргументот (низата `res`) претставува опис на поплочувањето. Во низата има $k \cdot l \cdot n$ знаци (без делимитер), кои претставуваат вредности `h` или `v`. Алгоритмот за креирање на `res` е:

- На почетокот нема знаци во `res`;
- Да претпоставиме дека одите по подот (правоаголната мрежа) во редици почнувајќи од првата до $l \cdot n$-вата, и за секој ред одиме по колони од првата до $k \cdot n$-вата. Ако застанете над горниот лев агол од некоја плочка, додавате еден знак во `res`: `h` – ако плочката е ставена во хоризонтален правец, или `v` – ако плочката е ставена во вертикален правец. Полињата кои не се горен лев агол од некоја плочка едноставно ги прескокнувате.

Ограничувања

Во 20% од тест случаите, $1 \le k, l \le 10$.

Во 30% од тест случаите, $n = 2$.

(Тест случаите во горните две множества делумно се поклопуваат).

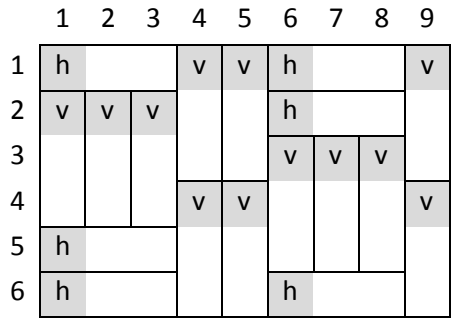
$1 \le k, l \le 57$, $2 \le n \le 10$.

Оценување

Ако недостасува или ако е неточен описот на регистрираното поплочување, за соодветниот тест случај не се доделуваат поени. На точен опис му се доделуваат поени според блискоста на бројот на поставените прашања q до теоретски потребниот број на прашања. Попрецизно, ако теоретскиот минимум е Q прашања и тест случајот е дизајниран за P поени, на точен опис ќе му се доделат $\min(P, P(Q/q)^{30})$ поени. Конечната сума се заокружува на најблискиот цел број.

Пример

Овој пример е направен според поплочувањето на сликата. Во сива боја се означени „почетоците“ на плочките, а буквата поставена во секој од „почетоците“ го означува правецот на плочката.



Ако, на пример, ги имате дефинирано променливите

```
int k,l,n;
```

тогаш повикот на функцијата

```
getArea(&k,&l,&n);
```

ќе ги постави вредностите на овие променливи, соодветно, на `k=3, l=2` и `n=3.`

Како илустрација за поставување на прашања, разгледајте го следниов фрагмент:

```
Data data[128] = {{1,1,0},{4,1,0},{7,1,0},{1,2,0},
                  {2,3,0},{3,4,0},{4,5,0},{5,6,0},{7,5,0}};
if (getData(9,data)) {// интерпретирај ги податоците вратени од
                      //  функцијата
}
else {//има грешка или ова не е првиот повик
}
```

Во разгледуваниот пример по првиот повик на функцијата **`getData`**, бидејќи податоците пред да се изврши повикот се во дозволените граници, низата `data` ќе изгледа вака:

```
{{x:1,y:1,d:'h'},{x:4,y:1,d:'v'},{x:6,y:1,d:'h'},
 {x:1,y:2,d:'v'},{x:2,y:2,d:'v'},{x:3,y:2,d:'v'},
 {x:4,y:4,d:'v'},{x:5,y:4,d:'v'},{x:7,y:3,d:'v'}}
```

Овој резултат треба да се регистрира со повик на `setResult`. Во овој пример, тоа би било:

```
char r[128]="hvvhvvvvhvvvvvvhhh";
setResult(r);
```

Коментар за примерот

Поставени се девет точни прашања (q = 9). Вратените девет точни одговори овозможуваат определување на поплочувањето, кое што е регистрирано во согласност со правилата. Бројот на поставени прашања го надминува теоретскиот минимум потребен за оваа конфигурација, кој што е Q = 6. Според тоа, на вакво решение би му биле доделени дел од дадените P поени, имено $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. Како што може и да претпоставите, одговор како овој, иако точен, практично би бил бескорисен.

Локално тестирање

За да можете да ја тестирате вашата функција *tiling* на вашиот локален компјутер, ќе ги добиете документите *Lgrader.cpp* и *tiling.h*. Компајлирајте го *Lgrader* заедно со вашиот документ **tiling.cpp** и ќе добиете програма која што можете да ја користите за тестирање на вашата функција.

Lgrader чита од стандарден влез во следниот формат:

- Линија 1 содржи три цели броеви: k, l и n.
- Следува матрица од цели броеви. Секој елемент на оваа матрица претставува цел број, кој означува реден број на плочка. Матрицата се состои од $l \cdot n$ линии; секоја линија содржи по $k \cdot n$ цели броеви.

Пример за локално тестирање (одговара на сликата погоре).

Влез	Излез
3 2 3 1 1 1 2 3 4 4 4 5 6 7 8 2 3 9 9 9 5 6 7 8 2 3 10 11 12 5 6 7 8 13 14 10 11 12 15 16 16 16 13 14 10 11 12 15 17 17 17 13 14 18 18 18 15	hvvhvvvvhvvvvvvhhh