КЛАРКСЪН

„Бийтбоксът на Джеръми Кларксън“ е популярен видеомонтаж в YouTube с бившия водещ на шоуто Top Gear, превърнато в бийтбокс. То се състои от поредица сцени от шоуто, подредени така, че да се получи забавна музикална компилация. Видеото, публикувано в YouTube през 2009 година, има почти три милиона гледания.

Тази година Джеръми Кларксън беше дисциплинарно уволнен от BBC. За да припомним култовото шоу, искаме да създадем следваща версия на популярния монтаж от YouTube, като използваме сцени от по-нови епизоди. Репликите за новото видео вече са избрани от привържениците на шоуто, но да се монтират не е толкова лесно.

Проблемът е там, че, независимо колко изкусно са свързани отделните части на видеото, преходите винаги са забележими за зрителя. Ако два прехода настъпят в кратък период от време, това може и да е дразнещо за някои зрители. Затова целта е да се поддържат преходите колкото е възможно по-отдалечени във времето един от друг чрез максимизиране на дължината на най-късата част от монтажа.

Задача

Напишете програма, която чете два реда от входа: текста на песента от първия ред и цитат от епизод на Top Gear от втория. Програмата трябва да намери най-добрия начин за съставяне на текста на песента от поднизове от цитата чрез максимизиране на дължината на най-късия подниз. Тя трябва да изведе дължината на най-късия подниз в този оптимален вариант на песента. Ако не е възможно съставяне на текста на песента от зададения цитат, програмата трябва да извежда −1.

Вход

Програмата чете два реда от стандартния вход, всеки от които е съставен само от малки и главни латински букви.

Изход

Програмата трябва да извежда на стандартния изход едно цяло число, равно на дължината на най-късия подниз при оптимален вариант на песента, или на -1, ако съставянето на песента е невъзможно.

Ограничения

Всеки от двата входни реда съдържа не повече от 100000 символа.

Пример

Вход

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

Изход

```
3
```

Обяснения

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

Подзадачи

Да означим дължината на първия входен низ с *N*, а на втория – с *M.*

Подзадача	Точки	*Max (N, M) е най-много*	Забележка
1	11	**10**	
2	17	**2 000**	
3	19	**10 000**	
4	23	**30 000**	Отговорът не надхвърля 20
5	30	**100 000**	

RADIO

Границата между България и Румъния е река Дунав (която може би вече сте видели). Възможно е тя да бъде премината с лодка и понякога хора преминават от едната държава в другата. За да няма инциденти, спасителната служба е построила по българския бряг на реката **N** охранителни кули. Техните позиции са на разстояние $\mathbf{X_1}$, $\mathbf{X_2}$, …, $\mathbf{X_N}$ метра по течението на реката, считано от мястото, където реката навлиза в България. За простота да си представим Дунав като отсечка, а кулите – като точки с целочислени координати по нея.

Във всяка кула *i* има радио-предавател с мощност $\mathbf{P_i}$, който може да комуникира с някои от останалите кули. Две кули *i* и *j* могат да комуникират помежду си, когато сумарната мощност на предавателите им е по-голяма или равна на разстоянието между тях, тоест: $|\mathbf{X_i} - \mathbf{X_j}| \leq \mathbf{P_i} + \mathbf{P_j}$.

За намаляване на разходите било взето решение някои от кулите да бъдат изведени от експлоатация. Планът е да бъдат оставени само **K** на брой от тези кули, а другите да бъдат продадени. Продаването на кула *i* носи на държавата $\mathbf{S_i}$ лева, но кулата вече не може да се използва.

Обаче, за по-добро охраняване, е въведено ново изискване − всеки две от оставените **K** кули да могат да комуникират директно помежду си. Тъй като това потенциално не е възможно със сегашните им предаватели, те трябва да бъдат модернизирани като се подобри тяхната мощност. За увеличаването на мощността на произволна кула с една единица е нужен един лев.

Задача

Ели наскоро започна стаж в министерството на финансите. Сега тя иска да блесне там, като предложи най-изгоден план за продажбата на $\mathbf{N} - \mathbf{K}$ кули и модернизиране на останалите, така че всеки две от тях да могат да комуникират директно помежду си (ако остане само една кула, не се интересуваме от комуникации). Вие решавате да ѝ помогнете, като напишете програма, която намира минималната цена (или максимална печалба, ако в резултат на продажбите бъдат спечелени повече пари, отколкото са изхарчени за обновяването) за извършване на модернизацията.

Вход

На първия ред на стандартния вход ще бъдат зададени две цели числа **N** и **K** – съответно броят кули в началото и в края на процеса на модернизация. Всеки от следващите **N** реда ще съдържа по три цели числа – $\mathbf{X_i}$, $\mathbf{P_i}$, $\mathbf{S_i}$ – съответно позицията на кулата, мощността на нейния предавател в началото и цената, за която може да бъде продадена. Кулите са зададени в нарастващ ред на позицията им $\mathbf{X_i}$. Никои две кули не се намират на една и съща позиция.

Изход

На единствен ред на стандартния изход изведете едно цяло число – минималната цена (или максималната печалба) за извършване на модернизацията. В случай на печалба, изведеното от Вас число трябва да е отрицателно.

Ограничения

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

Подзадачи

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

Примери

Вход	Вход
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

Изход	Изход
42	-24

Обяснение

В първия пример един оптимален вариант е да оставим кулите с номера {1, 3, 4}. За целта трябва да увеличим мощността на кула 1 със 17, а на кула 4 с 31, заплащайки 17 + 31 = 48 лева. От продажбата на кулите 2 и 5 печелим 4 + 2 = 6 лева. Така общо заплащаме 48 – 6 = 42 лева.

Във втория пример избираме, например, да запазим кулите с номера {2, 3, 6, 7, 9}. За да могат те да комуникират една с друга, увеличаваме мощността на 7 с 2, а тази на 9 с 4, за цена 2 + 4 = 6. При този избор на кули може да продадем кулите с номера {1, 4, 5, 8} за 4 + 6 + 9 + 11 = 30 лева, като сумарната "цена" е 6 – 30 = -24 лева. Така може да бъде постигната печалба в размер на 24 лева.

ОБЛИЦОВКА

Разглеждаме целите положителни числа *k*, *l* и *n*. Правоъгълна стая има размери на пода *k.n* × *l.n* единици. Подът е облицован с *k.l.n* на брой правоъгълни плочки с размери 1×*n*. Целостта на никоя плочка не е нарушавана при облицоването. Ще разглеждаме пода като мрежа с *k.n* колонки и *l.n* реда.

Ментално повреден робот се е заключил в стаята и заплашва да я взриви, ако не отгатнете точния начин на покриване на пода: как е разположена всяка плочка в облицовката. За щастие, роботът е склóнен да ви даде някаква информация, въз основа на която да разкриете точната облицовка. Вие можете да зададете набор от *q* въпроса от вида (*x y*), където $1 \le x \le k.n$ и $1 \le y \le l.n$ са номерá съответно на стълб и на ред (броенето започва от 1 от „горния ляв ъгъл“ – клетка в първа колона на първи ред).

Ще получите набор от *q* ***правилни*** отговора с точно каква плочка е покрита всяка от въпросните клетки: координатите на горния ляв ъгъл („началото“) на плочката и начина ѝ на поставяне („направлението“ ѝ) – „хоризонтално“ (по протежение на един ред) или „вертикално“ (върху един стълб).

Задача

Напишете функция `tiling()`, която ще бъде компилирана заедно с програма на журито. Функцията `tiling()` ще провежда кратък диалог с робота и от получената информация вие ще възстановявате облицовката.

Сигурно вече се гласите да попитате за всяка клетка и – готово. Уви, броят *q* на зададените въпроси трябва да е колкото може по-малък, иначе рискувате да подразните допълнително робота и въпреки, че сте разкрили правилната облицовка, да последва взрив (разбирайте – нула на теста: вижте начина на оценяване).

Детайли по реализацията

Вие трябва да предадете към системата програмен файл **tiling.cpp**, който съдържа функцията `tiling()`. Файлът може да съдържа и друг необходим код, но НЕ трябва да съдържа глобални декларации с имената, описани по-долу, нито името *main*. В началото си Вашият файл трябва да съдържа `#include "tiling.h"`.

Програмата на журито декларира и реализира 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);
```

Описание на дефинираните функции за комуникация с робота:

1. Функцията **`getArea`** ще върне параметрите на стаята *k*, *l* и *n* (броят колонки ще е *k.n*, а броят редове ще е *l.n*).
2. Вие можете да повикате **веднъж** функцията **`getData`** с описания по-горе прототип. Параметрите имат следния смисъл:

- *q* е броят на въпросите;
- входно-изходният масив от записи `quest` съдържа:
  - преди повикването – самите въпроси (*x* и *y* са съответно колонката и реда на клетката, за която задавате въпрос). Стойността на члена с име *d* при повикването няма значение;
  - след повикването – отговорите на робота, а именно: *x* и *y* са съответно колонката и реда на началото на плочката, покриваща съответната позиция от входа, за която сте попитали, а членът *d* е със стойност някой от двата символа: `h`, означаващ хоризонтално разположение на плочката, или `v`, който указва вертикално разположение. Ако функцията върне **`false`**, това значи, че или в някой от елементите на входно-изходния масив има некоректни входни данни (например, координати извън рамките на мрежата), или функцията вече е била повиквана. В този случай и целият входно-изходен масив ще се върне нулиран.

3. Накрая трябва да повикате дефинираната функция **`setResult`**, в която масивът от символи `res` описва облицовката. Това са *k.l.n* на брой символа (без разделител между тях), всеки от които `h` или `v`. Алгоритъмът на създаване на `res` е следният:

- отначало в `res` няма символи;
- представяте си, че обхождате пода (мрежата) по редове от първи до *l.n*-ти, а всеки от редовете по колонки от първа до *k.n*-та. Попаднете ли в горен ляв ъгъл на плочка, добавяте символ към `res`: `h`, ако тя е разположена хоризонтално, или `v`, ако е разположена вертикално. Клетките, които не са горен ляв ъгъл (начало) на плочка, просто се „прескачат“.

Ограничения

В 20% от тестовите примери $1 \le k,l \le 10$.

В 30% от тестовите примери $n = 2$.

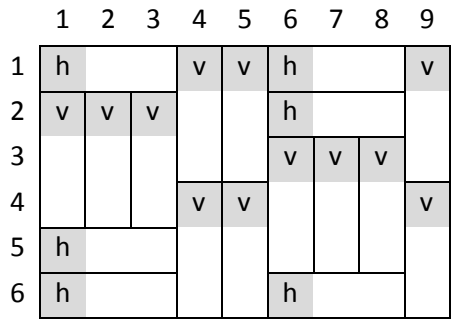
$1 \le k,l \le 57$, $2 \le n \le 10$.

Оценяване

При липса или регистрирано неправилно описание на облицовката съответният тестов пример не носи точки. Правилната облицовка носи точки в зависимост от близостта на броя зададени въпроси q до теоретично необходимия. По-точно, ако теоретично са необходими Q въпроса и за тестовия пример са предвидени P точки, при правилно изведена облицовка се получават $\min(P, P(Q/q)^{30})$ точки. Сумата от всички точки се закръгля до най-близкото цяло число.

Пример

Примерът отразява облицовката, показана на фигурата. Тук в сиво са означени началата на плочките, а с буква в тези начала е отбелязано направлението на всяка.



Ако, например, имате дефинирани променливи

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

то повикването на функция

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

ще установи стойностите на променливите съответно в `k=3, l=2` и `n=3.`

За примерно задаване на запитване вижте следния 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)) {//интерпретиране на получените данни}
else {//има грешка или повикването не е първо}
```

В разглеждания пример, след първо повикване на **`getData`**, тъй като преди повикването самите данни в него са в рамките на размерите, масивът `data` ще изглежда така:

```
{{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'}}
```

Накрая се регистрира резултат, чрез повикване на `setResult`. В този пример:

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

Коментар на примера

Зададени са девет коректни запитвания (q=9). Получените девет коректни отговора позволяват определяне на облицовката, което е регистрирано по правилата. Броят зададени въпроси надхвърля теоретично минимално необходимия за тази конфигурация, който е Q=6. Затова такова решение би донесло част от предвидените за теста точки, а именно $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. Както може да съобразите, такова решение, макар и вярно, е практически безполезно.

Локално тестване

За да можете при желание да тествате функцията *tiling* на локалния компютър, ви се предоставят файловете *Lgrader.cpp* и *tiling.h*. Компилирайте *Lgrader* заедно с Вашия файл **tiling.cpp** и ще получите програма, която можете да ползвате за тестване.

Lgrader чете от стандартния вход в следния формат:

- Ред 1: три цели числа *k*, *l* и *n.*
- Следва матрица от цели числа. Числото във всяка клетка на матрицата означава номер на плочка. Матрицата има *l.n* реда и във всеки ред има *k.n* числа.

Пример за локално тестване (съответства на фигурата)

Вход	Изход
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