CLARKSON

"Jeremy Clarkson Beatbox" este un montaj video popular pe YouTube în care fosta gazdă a emisiunii Top Gear interpretează beatbox. Videoclipul este compus dintr-o serie de scene din emisiune, lipite împreună pentru a realiza un spectacol muzical de divertisment. Videoclipul a fost publicat pe YouTube în 2009 şi a fost vizualizat de aproape trei milioane de ori.

Anul acesta Jeremy Clarkson a plecat de la BBC în urma unei hotărâri disciplinare. Pentru a sărbători emisiunea, noi vrem să producem o continuare a popularului montaj de pe YouTube folosind scene din cele mai recente episoade. Versurile noului videoclip sunt deja alese de fanii emisiunii, dar realizarea montajului nu este cel mai uşor lucru.

Problema este că, oricât de profesionist sunt lipite împreună părţile ce compun videoclipul, tranziţiile dintre părţile componente sunt mereu observate de privitor. Dacă două tranziţii apar într-o perioadă scurtă de timp, ele pot irita unii privitori. Prin urmare, scopul este de a păstra tranziţiile cât mai depărtate posibil prin maximizarea lungimii celei mai scurte părţi din montaj.

Cerinţă

Notă: În cele ce urmează, termenul de **subşir** semnifică o secvență de caractere aflate pe poziții consecutive.

Scrieţi un program care preia de la intrare două linii: versurile cântecului pe prima linie şi scenariul episodului din Top Gear pe a doua linie. Programul trebuie să determine cel mai bun mod de a alcătui versuri din subşirurile ce apar în scenariu pentru a maximiza lungimea celui mai scurt subşir. Programul trebuie să afişeze lungimea celui mai scurt subşir în varianta optimă a cântecului. Dacă este imposibil să se construiască cântecul episodului dat, trebuie să se afişeze -1.

Intrare

Programul vostru trebuie să citească de la intrarea standard două linii de text, care conţin doar litere mici şi mari ale alfabetului englez.

Ieşire

Programul vostru trebuie să afişeze la ieşirea standard un număr întreg, care este egal cu lungimea celui mai scurt subşir în varianta optimă a cântecului sau -1, dacă această constructie este imposibilă.

Restricţii

Fiecare linie ce compune datele de intrare va avea lungimea de cel mult 100000 de caractere.

Exemplu

Intrare

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

Ieşire

```
3
```

Explicaţie

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

Subtasks

Vom nota cu *N* lungimea primului şir, iar cu *M* lungimea celui de-al doilea şir.

subtask	points	*Max N, M is less than or equal to*	note
1	11	**10**	
2	17	**2 000**	
3	19	**10 000**	
4	23	**30 000**	The answer to the problem is less or equal to 20
5	30	**100 000**	

RADIO

După cum bine știți frontiera dintre Bulgaria și România este Dunărea.

Deoarece există posibilitate de a traversa râul cu barca, oamenii se foloseau de această posibilitate în trecut, dar și în prezent. Pentru a preveni accidentele, Serviciul de Salvare au construit pe malul bulgar **N** turnuri de securitate. Turnurile se află la o distanță de $\mathbf{X_1}$, $\mathbf{X_2}$, …, $\mathbf{X_N}$ metri între ele din punctul unde râul intră în Bulgaria. Pentru simplitate vom considera că Dunărea este sub forma unui segment și turnurile – puncte cu coordonate întregi pe acest segment.

Fiecare turn are câte un transmițător radio. Transmițătorul din turnul *i* are puterea $\mathbf{P_i}$. Cu ajutorul acestor transmițătoare turnurile comunică între ele. Două turnuri *i* și *j* pot comunica între ele dacă distanța dintre ele este mai mică sau egală cu suma puterilor transmițătorilor pe care le dețin (altfel spus dacă $|\mathbf{X_i} - \mathbf{X_j}| \leq \mathbf{P_i} + \mathbf{P_j}$).

Pentru a reduce costurile s-a luat decizia de a renunța la unele turnuri. Noul plan prevede că vor rămâne doar **K** turnuri iar restul vor fi vândute. Vânzarea turnului *i* va aduce guvernului $\mathbf{S_i}$ leva (lev-ul este moneda națională a Bulgariei), însă turnul nu va mai putea fi utilizat.

Oricum, pentru a păstra sistemul funcțional, a fost introdusă o nouă cerință: oricare două din cele **K** turnuri rămase trebuie să fie capabile să comunice între ele direct (atunci când doar un turn rămâne nu ne deranjează comunicarea). Deoarece aparent acest lucru este imposibil de realizat cu puterea lor actuală, există posibilitatea de a mări puterile turnurilor. Mărirea puterii unui turn cu o unitate de putere costă un lev.

Cerință

Domnișoara Elly și-a început recent practica de vară la Ministerul de Finanțe. Ea dorește să se evidențieze prin propunerea a celui mai rentabil plan pentru vânzarea celor **N – K** turnuri și mărirea puterii turnurilor rămase astfel încât orice pereche din turnurile rămase să poată comunica direct. Tu ai hotărât să o ajuți pe Elly elaborând un program care determină prețul minim (ori câștigul maxim dacă vânzarea turnurilor aduce mai mulți bani decât cheltuielile de modernizare a turnurilor rămase) pentru cerința dată.

Intrare

Pe prima linie a intrării standard sunt date două numere întregi **N** și **K** – numerele turnurilor la început și sfârșit. Pe următoarele **N** linii sunt trei numere întregi $\mathbf{X_i}$, $\mathbf{P_i}$, $\mathbf{S_i}$ – poziția turnului, puterea inițială a turnului și prețul cu care poate fi vândut. Turnurile sunt date în ordine crescătoare după pozițiile $\mathbf{X_i}$. Nu există două turnuri cu aceeași coordonată $\mathbf{X_i}$.

Ieșirea

Pe o singură linie a ieșirii standard, programul va afișa un număr întreg – prețul minim (ori câștigul maxim) pentru sarcina dată. În caz de câștig numărul tipărit va fi negativ.

Restricții

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

Subtask-uri

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

Exemple

Input	Input
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
Output	**Output**
42	-24

Explicații

În primul exemplu, o variantă optimală este de a păstra turnurile cu indicii {1, 3, 4}. Astfel este necesar să se mărească puterea turnului 1 cu 17 și puterea turnului 4 cu 31, achitând pentru asta 17+31=48 leva. Din vânzarea restului turnurilor (2 și 5) obținem 4 + 2 = 6 leva. Prețul total este 48 – 6 = 42 leva.

În al doilea exemplu putem alege, de exemplu, să păstrăm turnurile cu indicii {2, 3, 6, 7, 9}. Pentru ca turnurile să poată comunica fiecare între ele trebuie să mărim puterea turnului 7 cu 2 și puterea turnului 9 cu 4 pentru un preț de 2 + 4 = 6. Cu această mulțime de turnuri noi putem vinde {1, 4, 5, 8} pentru 4 + 6 + 9 + 11 = 30 leva. "Prețul" total este 6 – 30 = −24 leva. Astfel, avem câștig de 24 leva.

TILING

Fie numerele naturale *k*, *l*, şi *n*. O cameră dreptunghiulară are dimensiunile $k{\cdot}n \times l{\cdot}n$. Podeaua este pavată cu $k{\cdot}l{\cdot}n$ pavele dreptunghiulare de dimensiuni $1{\times}n$. Nicio pavelă nu a fost tăiată. Putem considera podeaua ca fiind o matrice cu $k{\cdot}n$ coloane şi $l{\cdot}n$ linii.

Un robot tulburat mental s-a închis în cameră şi ameninţă că o va arunca în aer dacă nu ghiciţi exact configuraţia pavajului: mai exact vrea să ştie poziţia fiecărei pavele. Din fericire, robotul este dispus să vă ofere nişte informaţii pe baza cărora să deduceţi această configuraţie. Puteţi pune robotului un număr de *q* întrebări de tipul *(x y)* unde $1 \le x \le k{\cdot}n$ şi $1 \le y \le l{\cdot}n$ sunt numerele unei coloane, respectiv ale unei linii (indexarea acestora se face începând cu 1 din colţul stânga sus – acesta este pe prima coloană şi prima linie).

Veţi primi un set de *q* răspunsuri **corecte** care relevă ce pavelă acoperă fiecare din celulele investigate: vi se vor oferi coordonatele colţului stânga sus al pavelei şi orientarea ei – "orizontală" (de-a lungul unei linii) sau "verticală" (de-a lungul unei coloane).

Cerinţă.

Scrieţi funcţia `tiling()`, care va fi compilată împreună cu programul comisiei şi care va realiza un scurt dialog cu robotul pentru a reconstitui configuraţia pavajului din informaţia primită.

Poate plănuiţi să puneţi câte o întrebare pentu fiecare celulă şi voilà! Totuşi, numărul *q* al întrebărilor ar trebui să fie cât mai mic, altfel riscaţi să enervaţi robotul şi ne vom confrunta cu o explozie chiar dacă aflaţi configuraţia corectă (acest scenariu se traduce printr-un punctaj de 0: vedeţi regulile de evaluare).

Detalii de implementare

Trebuie să trimiteţi spre evaluare un fişier **tiling.cpp,** care conţine funcţia `tiling()`. Fişierul poate conţine alte nume şi implementări necesare, dar nu are voie să folosească niciunul din numele de mai jos ca nume de variabile globale. Acest lucru este valabil şi pentru numele `main`.

La începutul fişierului trebuie să existe o linie care conţine directiva:

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

Programul comisiei declară şi implementează următoarele:

```
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);
```

Descrierea funcţiilor definite pentru comunicarea cu robotul:

1. Funcţia **`getArea`** va întoarce parametrii camerei *k*, *l* şi *n* (numărul de coloane va fi *k·n*, iar numărul de linii va fi *l·n*).
2. Puteţi apela **o dată** funcţia **`getData`** cu prototipul de mai sus. Parametrii au următoarele semnificaţii:

- *q* este numărul de întrebări;
- Vectorul de intrare/ieşire `quest` va conţine:
  - înainte de apel – întrebările (*x* şi *y* sunt linia, respectiv coloana pentru care se pune întrebarea). Valoarea membrului numit *d* este irelevantă aici.
  - după apel – răspunsurile robotului, anume: *x* şi *y* sunt coloana, respectiv linia colţului stânga sus al pavelei care acoperă celula respectivă iar membrul *d* are ca valoare unul din aceste două simboluri: `h`, semnificând direcţia orizontală, sau `v`, semnificând direcţia verticală. Dacă funcţia întoarce **`false`**, înseamnă că fie există date greşite în vectorul de intrare/ieşire (spre exemplu, indici care sunt în afara gridului) sau că funcţia a fost deja apelată. În acest caz, vectorul va conţine doar zerouri la ieşire.

3. Înainte de a se încheia apelul funcţiei `tiling()` aceasta trebuie să apeleze funcţia **`setResult`**, unde vectorul de simboluri `res` descrie configuraţia pavajului. Acestea sunt *k·l·n* simboluri (fără niciun separator), fiecare fiind `h` sau `v`. În continuare vom prezenta algoritmul prin care se obţine şirul `res`:

- Se începe fără niciun simbol în `res`;
- Imaginaţi-vă că parcurgeţi gridul linie cu linie, începând cu prima pâna la linia cu numărul *l·n* şi fiecare linie o parcurgeţi coloană cu coloană, începând cu prima până la coloana cu numărul *k·n*. Dacă păşiţi pe un colţ stânga sus al unei pavele, veţi adăuga un simbol la `res`: *h* pentru direcţia orizontală a pavelei sau *v* pentru direcţia verticală. Celulele care nu sunt colţuri stânga sus ale unei pavele sunt pur şi simplu ignorate.

Constrângeri

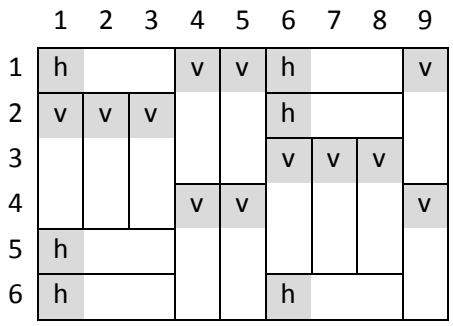
În 20% din teste, $1 \le k, l \le 10$.
În 30% din teste, $n = 2$.
$1 \le k, l \le 57$, $2 \le n \le 10$.

Punctaj

Dacă pentru un anumit test nu se oferă nicio configuraţie ca răspuns sau această configuraţie este greşită, punctajul pentru acest test va fi 0. O configuraţie corectă primeşte puncte în funcţie de apropierea numărului de întrebări puse, q, faţă de numărul minim necesar teoretic. Mai exact, dacă minimul teoretic este de Q întrebări iar punctajul testului este egal cu P, atunci o configuraţie corectă va primi $\min(P, P(Q/q)^{30})$ puncte. Suma finală va fi rotunjită la cel mai apropiat întreg.

Exemplu

Acest exemplu este bazat pe configuraţia din imagine. Sunt colorate cu gri capetele stânga sus ale pavelelor, iar fiecare celulă gri conţine o literă care denotă direcţia pavelei respective.



Dacă, spre exemplu, aţi definit variabilele

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

un apel al funcţiei

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

va schimba valorile acestor variabile în `k=3, l=2` şi `n=3.`

Pentru o ilustrare a modului de punere a întrebărilor, priviţi următorul fragment de cod:

```
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)) {//interpret the returned data}
else {//there is an error or this is not the first call}
```

În exemplul analizat, după primul apel al funcţiei **getData,** fiindcă datele sunt în limitele gridului, vectorul `data` va arăta în felul următor:

```
{{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'}}
```

Rezultatul trebuie înregistrat prin apelarea funcţiei `setResult`. În acest exemplu, să zicem:

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

Comentarii asupra exemplului

Au fost puse nouă întrebări corecte (q = 9). Cele nouă răspunsuri corecte oferite permit reconstituirea configuraţiei pavajului, care este înregistrată în concordanţă cu algoritmul descris. Numărul de întrebări puse depăşeşte minimul teoretic pentru acestă configuraţie, care este Q=6. Astfel, această soluţie ar primi o parte din cele P puncte, anume $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. După cum vă puteţi imagina, un asemenea răspuns, deşi corect, este practic inutil.

Testare locală

Pentru a vă putea testa funcţia *tiling* pe calculatorul vostru, veţi primi fişierele *Lgrader.cpp* şi *tiling.h.* Compilaţi *Lgrader* împreună cu fişierul vostru **tiling.cpp** și veți obține un program pe care-l puteți folosi pentru a vă testa funcția.

Lgrader citește datele de intrare în următorul format:

- Linia 1 conține trei numere întregi: k, l și n.
- Urmează o matrice de numere întregi. Fiecare întreg din acestă matrice reprezintă un număr de pavelă. Matricea are $l \cdot n$ linii, fiecare linie conținând $k \cdot n$ întregi.

Exemplu pentru testare locală (corespunde imaginii precedente).

Input	Output
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