CLARKSON

Džeremi Klarkson „beatbox" je popularna YouTube video montaža bivšeg Top Gear domaćina u obliku beatboxa. Video se sastoji od niza scena iz prethodnih emisija, montiranih kao jedan zabavni video klip. Objavljen je na YouTubeu 2009. godine, a do sada je pregledan gotovo tri milona puta.

Ove godine Džeremi Klarkson je otpušten sa BBC-a, nakon disciplinskog prekršaja. U spomen na kultni show, želimo proizvesti nastavak popularnog YouTube videa, montirajući scene iz novijih epizoda. Stihove za novi video već su izabrali fanovi, ali montaža nije tako jednostavna.

Problem je da, bez obzira na to koliko vješto spajate zasebne kadrove, prelazi (rezovi) su uvijek vidljivi gledaocima. Ako se dva prelaza smjenjuju u kratkom vremenskom razdoblju, to može biti iritantno za gledaoce. Dakle, cilj je zadržati prelaze na što je moguće većem rastojanju povećavanjem dužine najkraćeg kadrova u montaži.

Zadatak

Napiši program koji uzima dvije linije na ulazu: stihove pjesme na prvoj liniji, a tekst iz epizode Top Gear na drugoj liniji. Program zatim treba pronaći najbolji način za izgradnju stihova od podnizova teksta, povećavanjem dužine najkraćeg podniza. To bi trebalo dati dužinu najkraćeg podniza za optimalnu varijantu pjesme. Ako je nemoguće uvrstiti dio iz date epizode, na izlazu bi trebao dati -1. (Obavezno pogledaj primjer)

Input

Tvoj program da ulazu treba da dobije dvije linije teksta, koje sadrže samo engleska mala i velika slova.

Output

Tvoj program mora na standardnom izlazu dati jedan cijeli broj koji je jednak dužini najkraćeg podniza u optimalnoj varijanti pjesme ili -1 ako je konstrukcija nemoguća.

Ograničenja

Svaka linija na ulazu može imati najviše 100000 karaktera.

Primjer

Input

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

Output

```
3
```

Objašnjenje

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

Podzadatak

Označimo dužinu prvog stringa sa *N,* a dužinu drugog stringa sa *M*

podzadatak	poena	*Max N, M je manje ili jednako*	note
1	11	**10**	
2	17	**2 000**	
3	19	**10 000**	
4	23	**30 000**	Odgovor na problem je manje ili jednako 20
5	30	**100 000**	

RADIO

Granica između Bugarske i Rumunije je rijeka Dunav. U stvari, kao što ste već vjerovatno vidjeli – teško ju je promašiti.

Pošto je rijeku moguće preći čamcem, ljudi to često rade. U pokušaju da spriječe nesreće, Služba spasavanja je izgradila N sigurnosnih tornjeva na Bugarskoj obali. Njihove pozicije su udaljene $\mathbf{X_1}$, $\mathbf{X_2}$, …, $\mathbf{X_N}$ metara nizvodno od tačke gdje rijeka ulazi u Bugarsku. Zbog jednostavnosti mi ćemo pretpostaviti da je Dunav duž, a tornjevi – tačke sa cjelobrojnim koordinatama na njoj.

Na svakom tornju je radio predajnik. Predajnik na tornju *i* ima snagu $\mathbf{P_i}$. Koristeći ovaj predajnik tornjevi su u mogućnosti da komuniciraju međusobno. Dva tornja *i* i *j* mogu komunicirati ako i samo ako je udaljenost između njih manja ili jednaka od sume njihovih snaga predajnika (drugim riječima: ako je $|\mathbf{X_i} - \mathbf{X_j}| \leq \mathbf{P_i} + \mathbf{P_j}$).

U nastojanju da smanje troškove, donjeli su odluku da se uklone neki od tornjeva. Novi plan je da samo K tornjeva ostaje, a ostatak će biti prodat. Prodaja tornja *i* donosi vladi $\mathbf{S_i}$ EUR, ali nakon toga toranj naravno više ne može biti korišćen.

U svakom slučaju, da bi imali dobar funkcionalan sistem, mora biti ispunjeno da **svaka dva** preostala K tornja moraju biti u mogućnosti da komuniciraju međusobno direktno (kad ostane **samo jedan** toranj, smatramo da nema komunikacije). Pošto je vjerovatno nemoguće imati komunikaciju sa snagom preostalih tornjeva, oni moraju biti poboljšani. Poboljšanje tornja znači povećavanje njegove snage: za vrijednost jednog EUR-a dobija se jedna jedinica snage.

Zadatak

Gospođica Gabrijela je nedavno primljena kao pripravnica u Ministarstvo finansija. Ona sad želi da se pokaže u najboljem svjetlu predlažući najefikasniji plan za prodaju **N** – **K** tornjeva i poboljšanje preostalih tornjeva, tako da svaki par preostalih može komunicirati direktno između sebe. Ti si odlučio da pomogneš djevojci pišući program koji će naći minimalnu cijenu (ili maksimalnu zaradu, ako prodaja tornjeva donese više novca nego što će biti potrošeno na poboljšanja) za ovaj zadatak.

Ulaz

Na prvoj liniji standardnog ulaza, data su dva cijela broja N – broj tornjeva na početku i K - broj tornjeva na kraju, tim redom. Na svakoj od sljedećih N linija su data tri cijela broja X_i, P_i, S_i – pozicija odgovarajućeg tornja, njegova početna snaga i cijena za koju može biti prodat. Tornjevi su dati u rastućem poretku njihovih pozicija X_i. Ne postoje tornjevi sa istim koordinatama X_i. (i – je indeksirano počev od 1).

Izlaz

Na jedinoj liniji standardnog izlaza vaš program treba štampati jedan cijeli broj – minimalnu cijenu (ili maksimalnu zaradu) za dati zadatak. U slučaju zarade, štampani broj treba biti negativan.

Ograničenja

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

Podzadaci

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

Examples

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

Pojašnjenje

U prvom primjeru, jedna optimalna varijanta je zadržati tornjeve sa indeksima {1, 3, 4}. Na ovaj način moramo povećati snagu tornja 1 za 17 jedinica i snagu tornja 4 za 31 jedinicu, plaćajući 17 + 31 = 48 EUR-a. Prodajom preostalih tornjeva (2 i 5) zarađujemo 4 + 2 = 6 EUR-a. Ukupna cijena je 48 – 6 = 42 EUR.

U drugom primjeru možemo birati, na primjer, da zadržimo tornjeve sa indeksima {2, 3, 6, 7, 9}. U nastojanju da omogućimo tornjevima da i dalje komuniciraju međusobno mi moramo povećati snagu 7 tornja za 2 i snagu 9 tornja za 4, za cijenu 2 + 4 = 6 EUR. Sa ovim skupom tornjeva mi možemo prodati tornjeve {1, 4, 5, 8} za 4 + 6 + 9 + 11 = 30 EUR. Ukupna "cijena" je 6 – 30 = −24 EUR. Dakle imamo zaradu od 24 EUR-a.

TILING

Neka su k, l i n prirodni brojevi. Pod pravougaone sobe ima dimenzije $k{\cdot}n \times l{\cdot}n$ metara (tačka označava množenje, jedna dimenzija je $k{\cdot}n$ metara a druga $l{\cdot}n$ metara). Pod je popločan sa $k{\cdot}l{\cdot}n$ pravougaonih pločica dimenzija $1 \times n$. Nijedna pločica nije sječena prilikom popločavanja. Pod možemo zamisliti kao tablu sa $k{\cdot}n$ kolona i $l{\cdot}n$ redova.

Mentalno hendikepiran Petar se zaključao u gore pomenutu sobu i prijeti da će da je raznese ako ne pogodite kako je pod popločan tj. tačnu poziciju svake pločice u popločavanju. Srećom, Marko je odlučio da vam daje neke informacije na osnovu kojih možete pokušati da pogodite tačan izgled popločavanja. Možete postaviti (odjednom) q pitanja oblika (x, y) gde su $1 \le x \le k{\cdot}n$ i $1 \le y \le l{\cdot}n$ redni brojevi, kolone i reda, tim redom (brojanje počinje od 1 od "gornjeg lijevog ugla"– polja koje je u prvoj koloni i prvom redu).

Posle postavljanja svih pitanja, dobićete q ***tačnih*** odgovora – informacije koja pločica pokriva svako od upitanih polja tj. koordinate gornjeg lijevog ugla ("početka") pločice kao i informaciju da li je u "horizontalnoj" poziciji (duž reda) ili u "vertikalnoj" poziciji (duž kolone).

Zadatak

Napišite funkciju `tiling()`, koja će biti kompajlirana sa žirijevim programom i koja će voditi kratak dijalog sa Markom sa ciljem da rekonstruiše popločavanje na osnovu dobijenih informacija.

Možda planirate da postavite pitanje za svako polje i zeznete Marka. Međutim, broj pitanja q treba biti što manji mogući jer će se inače Marko iznervirati, pa će iako će vaš program pogoditi tačno popločavanje, Marko raznijeti sobu (tj. – dobićete nula poena za taj test primjer; vidi način bodovanja).

Detalji implementacije

Potrebno je sistemu za evaluaciju poslati fajl **tiling.cpp**, koji sadrži funkciju `tiling()`. Vaš fajl može sadržati dodatne funkcije i promenljive, ali obratite pažnju da globalna imena ne budu neka od onih navedenih ispod (u tiling.h fajlu) kao i da vaš fajl ne sadrži funkciju `main`.

Na početku fajla je neophodno da imate liniju sa sledećom preprocesorskom direktivom

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

Žirijev program deklariše i implementira sledeće stvari (struktura *Data* je data u tiling.h, nemojte je sami deklarisati):

```
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);
```

Opisi definisanih funkcija za komunikaciju sa Markom:

1. Funkcija **`getArea`** će vratiti parametre sobe *k*, *l* i *n* (broj kolona će biti *k*·*n*, a broj redova će biti *l*·*n*).
2. Funkciju **`getData`** možete pozvati **samo jednom** koristeći opisane parametre koji imaju sledeće značenje
   - *q* je broj pitanja koje želite da postavite;
   - ulazno/izlazni niz struktura `quest` sadrži:
     - prije poziva – vaša pitanja (*x* i *y* su, redom, kolona i red polja iz odgovarajućeg upita). Vrijednost člana strukture *d* tada nije bitno;
     - poslije poziva – Markove odgovore: *x* i *y* su, redom, kolona i red početka (gornjeg lijevog polja) pločice koja pokriva polje iz odgovarajućeg upita dok sada član strukture *d* ima jednu od sledeće dvije vrijednosti: `h`, koja označava horizontalnu pločicu, ili `v`, koja označava vertikalnu pločicu. Ako funkcija vrati **`false`**, to znači da ulazno/izlazni niz sadrži nekorektne podatke (npr. redni brojevi kolone/vrste su van granica table) ili je funkcija već bila pozvana od strane vašeg programa. **U tom slučaju će niz sadržati nule kao izlazne vrijednosti**.
3. Prije završetka, funkcija `tiling()` treba pozvati funkciju **`setResult`**, pri čemu niz karaktera `res` opisuje popločavanje. Ovaj niz treba da sadrži *k*·*l*·*n* karaktera (bez razmaka), od kojih je svaki ili `h` ili `v`. Evo algoritma za kreiranje niza `res`:

- Počnite sa praznim nizom `res`;
- Zamislite kao da šetate podom (tablom) red-po-red od prvog do *l*·*n*-tog, dok u svakom redu idete kolonama od prve do *k*·*n*-te. Ako stanete na gornje lijevo polje neke pločice, tada dodate jedan karakter u niz `res`: `h` za horizontalnu pločicu, odnosno `v` za vertikalnu pločicu. Polja koja nisu gornja lijeva polja (početna polja) neke pločice se jednostavno ignorišu.

Ograničenja

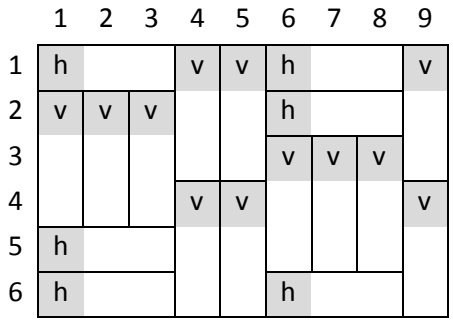
U 20% test primjera važi $1 \le k, l \le 10$.
U 30% test primjera važi $n = 2$.
Prethodna dva skupa test primjera ne moraju biti disjunktna.
$1 \le k, l \le 57$, $2 \le n \le 10$.

Bodovanje

Ako vaša funkcija ne vrati opis popločavanja ili vrati pogrešno popločavanje, na tom test primjeru dobijate 0 poena. U slučaju ispravnog opisa popločavanja, za dati test primjer dobijate poene na osnovu broja vaših pitanja q i na osnovu teoretski najmanjeg mogućeg broja pitanja za dati test primjer. Preciznije, ako je za dati test primjer teoretski minimalan broj pitanja Q a primjer vredi P poena, vaše rješenje će dobiti $\min(P, P(Q/q)^{30})$ poena. Finalni zbir poena po test primjerima biće zaokružen na najbliži cio broj.

Primjer

Primjer se odnosi na popločavanje na slici. Sivom bojom su označena početna bolja svake pločice dok slovo označava da li se radi o vertikalnoj ili horizontalnoj pločici. .



Ukoliko ste, na primjer, definisali promenljive

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

tada će poziv funkciji

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

postaviti vrijednosti ovih promjenljivih na `k=3, l=2` i `n=3.`

Za primjer slanja skupa pitanja, pogledajte sledeći kod:

```
 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)) {//radi nešto sa povratnim
vrijednostima}
 else {// postoji greška ili ovo nije prvi poziv}
```

U datom primjeru, posle poziva funkciji **`getData`** (ulazni niz je ispravan) niz `data` će izgledati ovako:

```
{{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'}}
```

Rezultat mora biti poslat uz pomoć funkcije `setResult`. U ovom primjeru to se može uraditi ovako:

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

Objašnjenje primjera

Postavljeno je 9 pitanja (q=9). Dobijenih 9 odgovora omogućuju da se odredi popločavanje a samo popločavanje je predstavljeno u nizu `res` u skladu sa pravilima. Postavljeni broj pitanja je veći od teoretski minimalnog broja pitanja da bi se riješila tabla ovih dimenzija koji je Q=6. Prema tome, ukoliko ovaj test primjer vrijedi P poena, ovo rješenje bi donijelo $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. Primjetimo da je ovakvo rješenje, iako tačno, praktično beskorisno što se broja osvojenih poena tiče.

Lokalno testiranje

Da biste mogli testirati funkciju *tiling* na lokalnom računaru, dobićete fajlove *Lgrader.cpp* i *tiling.h*. Kompajlirajte *Lgrader* zajedno sa vašim fajlom **tiling.cpp** i dobićete program koji možete koristiti za testiranje vaše funkcije.

Lgrader čita sa standardnog ulaza u sledećem formatu:

- Linija 1 sadrži tri cijela broja: k, l i n.
- Zatim slijedi matrica sa $l \cdot n$ redova i $k \cdot n$ kolona koja opisuje tablu. Broj u svakom polju označava redni broj pločice – susjedni isti brojevi čine jednu pločicu.

Primjer lokalnog testiranja (odgovara gornjem primjeru).

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