# CLARKSON

"Jeremy Clarkson Beatbox" je popularna YouTube video montaža bivšeg voditelja emisije Top Gear u obliku beatboxa. Video se sastoji od sekvence scena iz prethodnih emisija koje su montirane tako da čine jedan jako zabavan video snimak. Video je objavljen na sajtu YouTube 2009. godine i do sada ima skoro tri miliona pregleda.

Ove godine je Jeremy Clarkson otpušten sa BBC-a nakon disciplinskog prekršaja. Kako bi odali počast kultnoj emisiji, mladi producenti Didi i Midža žele da naprave *a sada drugi deo* popularne YouTube montaže, koristeći scene iz skorijih emisija. Tekst nove montaže je unapred izabran od strane fanova emisije, ali pravljenje montaže nije lak posao.

Problem je u tome što bez obzira koliko vešto mladi producenti montiraju scene, prelazi su uvek vidljivi gledaocima. Ako se dva prelaza dese u bliskim vremenskim trenucima to može biti veoma iritantno gledaocima. Dakle, njihov cilj je da prelaze razmaknu što je moguće više, tako što će maksimizovati dužinu najkraće scene u montaži. Za detaljnije objašnjenje cilja ovih mladih kompozitora **pogledati objašnjenje test primera**.

## Zadatak

Vaš zadatak je da napišete program koji će pomoći mladim kompozitorima da naprave najbolju montažu na svetu. Vaš program kao ulaz prima dva stringa: tekst nove montaže u prvoj liniji i scenario emisije Top Gear u drugoj liniji. **Vaš program treba da pronađe najbolji način da konstruiše tekst nove montaže (tj. prvi string) koristeći scene emisije (tj. koristeći podstringove drugog stringa) tako što će maksimizovati dužinu najkraće scene (tj. najkraćeg podstringa)**. Vaš program treba da ispiše upravu tu optimalnu dužinu. Ukoliko nije moguće konstruisati tekst nove montaže iz datog scenarija, ispisati -1.

## Input

Vaš program treba da pročita sa standardnog ulaza dva stringa koji se sastoje samo od velikih i malih slova engleske abecede.

## Output

Vaš program treba da ispiše na standardni izlaz jedan ceo broj koji je jednak dužini najmanjeg podstringa u optimalnoj montaži, ili -1 ukoliko sklapanje montaže nije moguće.

## Ograničenja

Oba stringa će imati dužinu manju ili jednaku 100000 karaktera.

## Test primer

### Ulaz

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

### Izlaz

```
3
```

## Objašnjenje primera

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

## Subtasks

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

| 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** | Rešenje zadatka nije veće od dvadeset |
| 5 | 30 | **100 000** | |

# RADIO

Granica između Bugarske i Rumunije je reka Dunav. Zapravo, verovatno ste je već videli – jako je teško ne videti je.

Pošto je reku moguće preći čamcem, ljudi to rade veoma često. Kako bi se nezgode sprečile, obalska straža je sagradila **N** bezbednosnih tornjeva duž obale Bugarske. Tornjevi se nalaze na udaljenostima $\mathbf{X_1}$, $\mathbf{X_2}$, …, $\mathbf{X_N}$ metera nizvodno od mesta na kojem Dunav ulazi u Bugarsku. Radi jednostavnosti Dunav ćemo posmatrati kao duž, a tornjeve kao tačke sa celobrojnim koordinatama koje pripadaju toj duži.

Unutar svakog tornja se nalazi radio predajnik. Radio predajnik unutar tornja *i* ima snagu $\mathbf{P_i}$. Tornjevi koriste radio predajnike za međusobnu komunikaciju. Dva tornja *i* i *j* mogu da komuniciraju ako i samo ako je njihova udaljenost manja ili jednaka sumi snaga njihovih radio predajnika. (drugim rečima: ako i samo ako važi $|\mathbf{X_i} - \mathbf{X_j}| \leq \mathbf{P_i} + \mathbf{P_j}$).

Kako bi se smanjili režijski troškovi doneta je odluka da se određen broj tornjeva proda, a da se zadrži samo **K** tornjeva. Prodaja *i-tog* tornja donosi vladi $\mathbf{S_i}$ dinara, ali se toranj ne može više koristiti.

Međutim, kako bi novi sistem bio funkcionalan, donet je zakon da **svaka dva** od preostalih **K** tornjeva moraju biti u mogućnosti da komuniciraju međusobno (ne marimo za komunikacije kada **samo jedan** toranj ostane). Kako je to potencijalno nemoguće sa trenutnim snagama radio predajnika mi ih moramo unaprediti. Cena povećanja snage radio predajnika za jednu jedinicu iznosi jedan dinar.

## Zadatak

Gospodin Zdule je nedavno postavljen za ministra finansija. On sada želi da se dokaže tako što će predložiti najisplativiji plan za prodaju **N** − **K** tornjeva i potencijalno unapređivanje ostalih, tako da svaki par preostalih tornjeva može da komunicira međusobno. Kako Zdule trenutno pomaže svom drugu Marku da poploča svoju sobu, on vas moli da mu pomognete tako što ćete napisati program koji će pronaći minimalnu cenu (ili maksimalnu dobit, ako prodaje tornjeva donosi više dinara nego što je potrebno za njihovo unapređivanje) potrebnu da bi sistem bio funkcionalan.

## Ulaz

Prva linija standardnog ulaza sadrži dva cela broja **N** i **K** – ukupan broj tornjeva i broj tornjeva koje želimo da zadržimo, tim redom. Svaka od sledećih **N** linija sadrži tri cela broja $\mathbf{X_i}$, $\mathbf{P_i}$, $\mathbf{S_i}$ – poziciju tornja, početnu snagu i njegovu prodajnu vrednost, tim redom. Tornjevi su rastuće sortirani po poziciji $\mathbf{X_i}$. Nikoja dva tornja nemaju jednake koordinate $\mathbf{X_i}$.

## Izlaz

Na prvoj i jedinoj liniji standardnog izlaza vaš program treba da ispiše jedan ceo broj – minimalni trošak (ili makisimalnu dobit) potreban da sistem bude funkcionalan. U slučaju dobiti ispisani broj treba biti negativan.

## Ograničenja

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

## Subtasks

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

## Primeri

| Ulaz | Ulaz |
|---|---|
| 5 3<br>4 63 3<br>13 2 4<br>87 3 9<br>121 6 15<br>159 5 2 | 9 5<br>5 8 4<br>10 10 7<br>11 9 7<br>13 6 6<br>19 20 9<br>20 2 1<br>23 1 3<br>26 13 11<br>28 4 2 |

| Izlaz | Izlaz |
|---|---|
| 42 | -24 |

## Objašnjenje test primera

U prvom primeru, jedno optimalno rešenje je da zadržimo tornjeve sa indeksima {1, 3, 4}. Sada moramo povećati snagu prvog tornja za 17 i snagu četvrtog tornja za 31 po ceni 17 + 31 = 48 dinara. Prodajom ostalih tornjeva (drugog i petog) zarađujemo 4 + 2 = 6 dinara. Ukupan trošak je 48 – 6 = 42 dinara.

U drugom primeru, jedno optimalno rešenje je da sadržimo tornjeve sa indeksima {2, 3, 6, 7, 9}. Sada moramo povećati snagu sedmog tornja za dva i snagu devetog tornja za četiri po ceni 2 + 4 = 6. Prodajom ostalih tornjeva (prvog, četvrtog, petog i osmog) zarađujemo 4 + 6 + 9 + 11 = 30 dinara. Ukupan "trošak" je 6 – 30 = −24 dinara. Dakle, imamo zaradu od 24 dinara.

# 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 sečena prilikom popločavanja. Pod možemo zamisliti kao tablu sa $k{\cdot}n$ kolona i $l{\cdot}n$ redova.

Mentalac Marko se zaključao u gore pomenutu sobu i preti 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 levog 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. koodinate gornjeg levog 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 i, iako će vaš program pogoditi tačno popločavanje, Marko će razneti sobu (tj. – dobićete nula poena za taj test primer; videti 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:
     - pre poziva – vaša pitanja (*x* i *y* su, redom, kolona i red polja iz odgovarajućeg upita). Vrednost člana strukture *d* tada nije bitno;
     - posle poziva – Markove odgovore: *x* i *y* su, redom, kolona i red početka (gornjeg levog polja) pločice koja pokriva polje iz odgovarajućeg upita dok sada član strukture *d* ima jednu od sledeće dve vrednosti: `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 vrednosti**.
3. Pre 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 levo polje neke pločice, tada dodate jedan karakter u niz `res`: `h` za horizontalnu pločicu, odnosno `v` za vertikalnu pločicu. Polja koja nisu gornja leva polja (početna polja) neke pločice se jednostavno ignorišu.

## Ograničenja

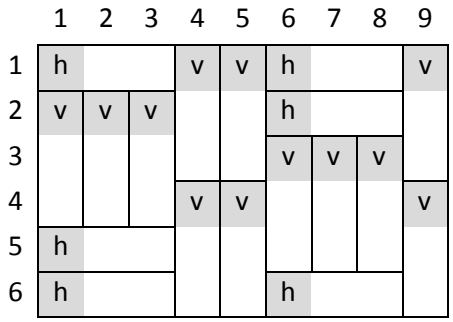
U 20% test primera važi $1 \le k, l \le 10$.
U 30% test primera važi $n = 2$.
Prethodna dva skupa test primera ne moraju biti disjunktna.
$1 \le k, l \le 57$, $2 \le n \le 10$.

## Bodovanje

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

## Primer

Primer 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 primer, definisali promenljive

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

tada će poziv funkciji

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

postaviti vrednosti ovih promenljivih na `k=3, l=2` i `n=3.`

Za primer 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 vrednostima}
else {// postoji greška ili ovo nije prvi poziv}
```

U datom primeru, 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 primeru to se može uraditi ovako:

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

## *Objašnjenje primera*

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 rešila tabla ovih dimenzija koji je Q=6. Prema tome, ukoliko ovaj test primer vredi $P$ poena, ovo rešenje bi donelo $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. Primetimo da je ovakvo rešenje, 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 cela broja: $k$, $l$ and $n$.
- Zatim sledi matrica sa $l \cdot n$ redova i $k \cdot n$ kolona koja opisuje tablu. Broj u svakom polju označava redni broj pločice – susedni isti brojevi čine jednu pločicu.

*Primer lokalnog testiranja (odgovara gornjem primeru).*

| Input | Output |
|---|---|
| 3 2 3<br>1 1 1 2 3 4 4 4 5<br>6 7 8 2 3 9 9 9 5<br>6 7 8 2 3 10 11 12 5<br>6 7 8 13 14 10 11 12 15<br>16 16 16 13 14 10 11 12 15<br>17 17 17 13 14 18 18 18 15 | hvvhvvvvhvvvvvvhhh |