CLARKSON

"Jeremy Clarkson Vidyosu" YouTube'da oldukça popüler olan ve eski bir program sunucusu olan Jeremy Clarkson'un yaptığı ilginç bir vidyodur. Video, Jeremy Clarkson'ın programından bir çok sahne içermektedir. Bu sahneler birbiri ile birleştirilerek eğlenceli bir şekle getirilmiştir. Video, YouTube'da 2009 yılında yayınlanmıştır ve bu güne kadar milyonlarca kişi tarafından seyredilmiştir.

Bu yıl, Jeremy Clarkson'un BBC'deki işine son verilmiştir. Bu programı yad etmek için programın son bölümlerini kullanarak önceki YouTube vidyosuna benzer bir montaj video oluşturmak istiyoruz. Bu yeni vidyonun şarkı sözleri programın hayranları tarafından seçilmiştir ancak montajlama sanıldığı kadar kolay bir işlem değildir.

Vidyonun farklı parçaları becerikli bir şekilde birleştirilse bile, parçalar arasındaki geçişlerin izleyiciler için fark edilmesi büyük bir problem oluşturmaktadır. Eğer iki geçiş kısa bir süre içinde olursa, bu bazı izleyiciler için sinir bozucu olabilmektedir. Bu nedenle, amacımız montajdaki en küçük parça uzunluğunu maksimize ederek geçişleri mümkün olduğunca birbirinden ayırmaktır.

Görev

İki satır alan bir kod yazın: İlk satırda şarkının sözleri, ve ikinci satırda programın bir bölümünün seslendirmesi olacaktır. Programınız daha sonra, ikinci satırda verilen bölüm seslendirmesinin alt dizgilerini kullanarak, ilk satırdaki şarkı sözünü elde etmelidir. Bu işlem, şarkı sözünün en küçük alt dizgi uzunluğu maksimize edilecek şekilde yapılmalıdır. Program, şarkı sözünün optimal olarak bulduğu en küçük alt dizgisinin uzunluğunu çıktı olarak vermelidir. Eğer şarkı, verilen bölüm seslendirilmesinden oluşturulamazsa programınız çıktı olarak -1 vermelidir.

Girdi

Programınız standart girdiden iki satırlık metin okumalıdır, bu girdiler sadece büyük ve küçük İngilizce harfleri olacaktır.

Çıktı

Programınız standart çıktıya bir tane tam sayı yazmalıdır. Bu sayı, şarkı sözünün optimal olarak bulduğu en küçük alt dizgisinin uzunluğuna eşit olacaktır. Veya eğer şarkı, verilen bölüm seslendirilmesinden oluşturulamazsa bu sayı -1 olacaktır.

Sınırlar

Girdideki her satır en fazla 100000 karakter uzunluğunda olacaktır.

Örnek

Girdi

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

Çıktı

```
3
```

Açıklama

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

Altgörevler

İlk dizginin uzunluğunu *N* ile, ve ikinci dizginin uzunluğunu *M* ile gösterelim.

altgörev	puan	*Maksimum N, M küçük eşit*	not
1	11	**10**	
2	17	**2 000**	
3	19	**10 000**	
4	23	**30 000**	Problemin cevabı küçük eşit 20
5	30	**100 000**	

RADYO

Tuna nehri Bulgaristan ve Romanya arasında sınırdır. Tuna'yı henüz görmediyseniz aşkolsun size!

Geçmişte ve günümüzde nehri tekneyle geçmek mümkündür. Kazaları önlemek için Kurtarma Teşkilatı nehrin Bulgar kıyısı boyunca **N** tane güvenlik kulesi inşaa etmiştir. Kulelerin konumları nehrin Bulgaristan'a girdiği noktadan itibaren X_1, X_2, …, X_N metre aşağıdadır. Basitlik açısından Tuna nehrini düz bir çizgi ve kuleleri ise bu çizgi üzerinde tamsayı koordinatlı noktalar olarak düşüneceğiz.

Kulelerde birer tane radyo vericisi vardır. Kule *i* deki vericinin gücü P_i dir. Bu vericiler vasıtasıyla kuleler birbiri ile haberleşme şansına sahiptir.

i ve *j* kuleleri eğer aralarındaki mesafe vericilerinin güçleri toplamından küçük eşitse haberleşebilir. Yani, *i* ve *j* kulelerinin haberleşebilmesi için $|X_i - X_j| \leq P_i + P_j$ olması gerekir.

Maliyetleri azaltmak için bazı kuleleri devre dışına alacak bir karar verilmiştir. Karara göre sadece **K** tane kule faal kalacak diğerleri ise satılacaktır. Kule *i* yi satmak hükümete S_i leva gelir getirecektir. Satılan kuleler artık kullanım dışıdır.

İyi çalışan bir sistem olması açısından, yeni bir gereksinim de ortaya çıkmıştır: Faal kalacak olan **K** kule birbiri ile doğrudan haberleşebilmelidir (tek bir kule faal kaldıysa haberleşme derdi zaten yoktur). Vericilerin ilk güçleri ile kulelerin karşılıklı doğrudan haberleşmesi mümkün olamayabileceğinden vericilere güç takviyesi gerekecektir. Herhangi bir vericinin 1 birim güç takviyesinin bedeli 1 leva'dır.

Görev

Elly, Maliye Bakanlığı'nda yaz stajına başlar başlamaz bu işi ona kitlemişlerdir. Elly kendini göstermek için **N** − **K** adet kuleyi satıp geri kalan **K** kuledeki vericileri takviye ederek birbiri ile haberleşmesini sağlayacak en maliyet-etkin çözüm üzerinde düşünür. Sizde O na yardımcı olmak istiyorsunuz. Sizden istenen minimum masrafla (başka bir deyişle maksimum kazanç, çünkü satılan kulelerin toplam fiyatı faal kalan vericilerin takviye masrafından büyük olabilir) bu işi çözecek programı yazmanızdır.

Girdi

Standart girdi de ilk satırda iki tamsayı **N** ve **K** olacaktır – sırasıyla başlangıçtaki kule sayısı ve sonuçta faal kalacak kule sayısı. Takip eden **N** satırın herbirinde üç tamsayı $\mathbf{X_i}$, $\mathbf{P_i}$, $\mathbf{S_i}$ olacaktır– sırasıyla kulenin konumu, kuledeki vericinin ilk gücü ve kulenin satış fiyatı. Kuleler artan $\mathbf{X_i}$ konumlarına göre sıralı verilecektir. Aynı $\mathbf{X_i}$ konumunda birden fazla kule yer almayacaktır.

Çıktı

Standart çıktıya tek bir tamsayı yazdırın- minimum masraf (maksimum kazanç) değeri. Kazanç durumunda yazdıracağınız tamsayı negatif olmalıdır.

Kısıtlar

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

Altgörevler

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

Örnekler

Girdi	Girdi
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
Çıktı	**Çıktı**
42	-24

Açıklama

İlk örnekte, bir optimal çözüm {1, 3, 4} numaralı kuleleri faal tutmaktır. Böylelikle, kule 1'in gücünü 17 birim ve kule 4'ün gücünü 31 birim takviye ederek 17 + 31 = 48 leva öderiz. Geri kalan iki kuleyi (2 ve 5) satarak 4 + 2 = 6 leva kazanırız. Minimum masraf 48 – 6 = 42 leva'dır.

İkinci örnekte, {2, 3, 6, 7, 9} numaralı kuleleri faal bırakmayı seçebiliriz. Faal kulelerin birbiri ile doğrudan karşılıklı haberleşebilmeleri için kule 7'nin gücünü 2 birim ve kule 9'un gücünü 4 birim takviye ederek 2 + 4 = 6 leva masraf ederiz. Geri kalan {1, 4, 5, 8} numaralı kuleleri satarak 4 + 6 + 9 + 11 = 30 leva kazanırız. Böylelikle maksimum kazanç 24 leva olup cevabımız 6 – 30 = −24'tür.

FAYANS DÖŞEME

Üç tane pozitif tam sayı k, l, ve n verilmiş olsun. Bir dikdörtgen odanın zemininin boyutları da $k \cdot n \times l \cdot n$ birim olsun. Odanın zemini, her biri $1 \times n$ boyutlarında $k \cdot l \cdot n$ adet fayans ile döşenmiştir. Döşeme sırasında hiçbir fayans parçalanmamıştır. Odanın zeminini $k \cdot n$ sütun ve $l \cdot n$ satırdan oluşan bir ızgara olarak düşünebilirsiniz.

Kafayı yemiş bir robot kendini bu odaya kilitlemiş ve eğer fayansların tam olarak ne şekilde döşendiğini bilemezseniz odayı havaya uçuracağını söylüyor. Yani zeminde her bir fayansın tam olarak nereye konduğunu bulmanızı istiyor. Şansımıza, robot size bir takım bilgiler vermeye ikna olmuş. Bu bilgileri kullanarak fayansların nasıl döşendiğini bulmanız mümkün. Robota tek bir seferde, aşağıda anlatılan formatta, q tane soru sormanıza izin veriliyor. Sorular (x, y) şeklindedir ($1 \leq x \leq k \cdot n$ ve $1 \leq y \leq l \cdot n$) ve x bir sütunu, y de bir satırı ifade eder. Sütun ve satır sayıları üst sol köşede yani ilk sütun ve ilk satırda 1'den başlar.

Robot size q tane ***doğru*** cevap verecektir ve her bir cevap, sorgulanan hücrede hangi fayansın olduğunu belirtecektir. Verilen cevap, sorgulanan hücredeki fayansın sol üst köşesinin (yani başlangıç) koordinatlarını ve ne şekilde yerleştirildiğini (yani "yönünü") söyleyecektir – "yatay" (satır boyunca) ya da "dikey" (sütun boyunca).

Görev

`tiling()` adında bir fonksiyon yazacaksınız. Bu fonksiyon, jürinin programı ile birlikte derlenecektir ve robot ile ufak bir diyalog gerçekleştirerek robottan alınan bilgiler ile fayansların tam olarak nasıl döşendiğini bulacaktır.

Her bir hücre için bir soru sorarak görevi tamamlamak çok kolay olsa da sizden istenen, olabilecek en az sayıda soru sorarak görevi gerçekleştirmektir. Yani q sayısı en küçük olmalıdır. Yoksa fayansların yerini doğru bile bulsanız, fazla soru nedeniyle robot sinirlenip odayı patlatabilir (yani sorudan 0 puan alırsınız: Notlandırma Kurallarına bakınız).

Gerçekleştirim detayları

tiling.cpp adında `tiling()` fonksiyonunu içeren bir dosya göndermelisiniz. Dosyanızda başka fonksiyonlar ya da tanımlar bulunabilir ama aşağıda verilen değişken isimleri ya da `main` fonksiyonu bulunmaMAlıdır.

Dosyanızın başında aşağıdaki satır bulunmalıdır:

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

Jürinin programı aşağıdakileri tanımlar ve gerçekleştirir:

```
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);
```

Robot ile iletişim için tanımlanan fonksiyonların açıklaması:

1. **`getArea`** fonksiyonu oda parametrelerini yani k, l ve n'yi döndürür (sütun sayısı $k \cdot n$ ve satır sayısı $l \cdot n$ olur) .
2. **`getData`** fonksiyonunu yukarıdaki prototipteki parametreler ile yalnızca **bir kez** çağırabilirsiniz. Parametreler aşağıda açıklanmıştır:

- q soru sayısıdır;
- Hem girdi hem de çıktı olarak kullanılan kayıt dizisi `quest` aşağıdaki bilgileri içerir:
  - fonksiyon çağrılmadan önce – soruları içerir (x ve y sorgulanan hücrenin sütun ve satırını belirtir). Kayıt içindeki *d* değeri burada önemsizdir;
  - fonksiyon çağrıldıktan sonra – robotun cevaplarını içerir, yani: x ve y sırasıyla sorgulanan hücredeki fayansın başlangıç hücresinin sütun ve satırını ve d değeri iki sembolden birini içerir: `h`, fayansın yatay döşendiğini, ve `v`, fayansın dikey döşendiğini gösterir. Girdi/çıktı dizisinde hatalı bir veri varsa (mesela sınırları aşan sorgular varsa), ya da fonksiyon ikinci kez çağrılmışsa, bu fonksiyon **`false`** döner. Bu durumda çıktı dizisi değer olarak hep sıfır içerir.

3. `tiling()` fonksiyonu sonlanmadan önce **`setResult`** fonksiyonunu çağırarak fayansların yerlerini belirtmelidir. Fayansların yerlerini belirtmek için `res` sembol dizisi kullanılır. `res` dizisinde $k \cdot l \cdot n$ sembol vardır (herhangi bir ayraç sembol olmadan) ve her bir sembol ya `h` ya da `v`'dir. Aşağıda, `res` dizisini oluşturmak için kullanılan algoritma açıklanmıştır:

- Boş bir `res` dizisi ile başlayın
- Zeminde satır satır yürüyün (1. satırdan $l \cdot n$-nci satıra kadar), ve her bir satırda da 1. sütundan $k \cdot n$-nci sütuna doğru yürüyün. Bu yürüme sırasında bir fayansın sol üst köşesine rastladığınızda eğer fayans yatay ise `res`'e `h` sembolünü dikey ise de `v` sembolünü ekleyin. Herhangi bir fayansın sol üst köşesi yani başlangıcı olmayan hücreler için `res`'e hiçbir ekleme yapmadan basitçe hücrenin üzerinden atlayarak yürümeye devam edin.

Kısıtlar

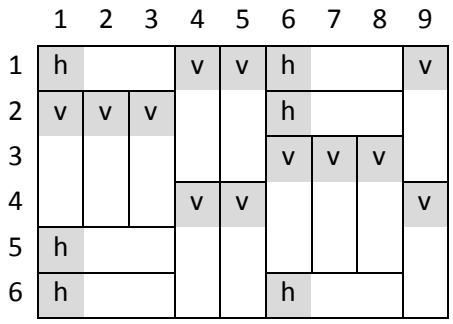
Testlerin %20'sinde, $1 \le k, l \le 10$.
Testlerin %30'unda, $n = 2$.
$1 \le k, l \le 57$, $2 \le n \le 10$.

Notlandırma

Eğer bir fayans için yanlış bir sonuç varsa ya da sonuçta eksik olan fayanslar varsa, programınız hiç puan alamaz. Doğru şekilde fayansların yerini bulan bir cevap sorulan soru sayısı olan q'nun teorik olarak en az sayıda gerekli olan soru sayısına yakınlığına göre puanlandırılır. Teorik minimum Q ise ve o test P puanlık bir test ise, q soru sorarak bulunmuş bir doğru fayans yerleşim cevabı $min(P, P(Q/q)^{30})$ puan alacaktır. Toplam alınacak puan en yakın tamsayıya yuvarlanacaktır.

Örnek

Bu örnek yandaki şekilde gösterilen fayans yerleşimine göre hazırlanmıştır. Gri hücreler fayans başlangıçlarını belirtmektedir, ve başlangıç hücresindeki harfler fayans yönünü göstermektedir.



Eğer aşağıdaki değişkenleri tanımlamışsanız

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

şu fonksiyon çağrısı

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

değişkenlerin değerlerini `k=3, l=2` ve `n=3` yapacaktır.

Soruların nasıl sorulacağı ile ilgili bir örnek için aşağıdaki kod parçasına bakabilirsiniz:

```
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)) {//donen veriyi incele}
else {//bir hata var ya da bu ilk fonksiyon cagrisi degil}
```

Bu örnekte **`getData`** fonksiyonuna yapılan ilk çağrıdan sonra veriler sınırların içinde olduğu için `data` adlı dizi aşağıdaki gibi görünecektir:

```
{{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'}}
```

Bulunan sonuç `setResult` fonksiyonu çağrılarak belirtilmelidir. Bu örnekte mesela:

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

Örnekle İlgili Yorumlar

Dokuz doğru soru sorulmuştur ($q = 9$). Dönen dokuz doğru cevap fayansların yerini bulmamıza yeterlidir ve cevap kurallara uygun bir şekilde belirtilmiştir. Sorulan soru sayısı teorik minimum olan, Q=6'dan fazladır. Yani böyle bir çözüm bu test için ayrılan *P* puanın {*P*(2/3)^{30}}≈{0.000005 *P*} kadarını alır. Tahmin edebileceğiniz gibi böyle bir cevap doğru da olsa pratikte hiç puan almaz.

Bilgisayarınızda test etme

tiling fonksiyonunu kendi bilgisayarınızda test etmek için *Lgrader.cpp* ve *tiling.h* dosyalarını almalısınız. *Lgrader*'I dosyanız **tiling.cpp** ile birlikte derleyerek test için kullanabileceğiniz bir program elde edebilirsiniz.

Lgrader standart girdiyi aşağıdaki formatta okur:

- Birinci satır üç tam sayı içerir: k, l ve n.
- Sırada bir tam sayı matrisi yer alır. Bu matrisin her bir hücresindeki tam sayı bir fayans numarasını belirtir. Matriste $l \cdot n$ satır ve her satırda $k \cdot n$ tamsayı vardır.

Bilgisayarınızda test için örnek (yukarıda verilen şekle karşılık gelen girdi/çıktı).

Girdi	Çıktı
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