CLARKSON

Το "Jeremy Clarkson Beatbox" είναι ένα δημοφιλές βίντεο μοντάζ στο YouTube που δείχνει τον τέως παρουσιαστή του Top Gear να τραγουδάει beatbox. Το βίντεο αποτελείται από μια σειρά από σκηνές του σόου, συνδεδεμένες μεταξύ τους ούτως ώστε να δημιουργούν μια μουσική παράσταση. Το βίντεο δημοσιεύτηκε στο YouTube το 2009 και έχει ήδη τρία εκατομμύρια προβολές.

Φέτος ο Jeremy Clarkson αποχώρησε από το BBC. Για να τον τιμήσουμε θέλουμε να δημιουργήσουμε τη συνέχεια του δημοφιλούς YouTube μοντάζ, χρησιμοποιώντας σκηνές από τα πιο πρόσφατα επεισόδια. Οι στίχοι του νέου βίντεο έχουν ήδη επιλεγεί από τους οπαδούς του σόου, αλλά η δημιουργία του μοντάζ δεν είναι εύκολη.

Το πρόβλημα είναι ότι όσο καλά κι αν συνδεθούν οι διάφορες σκηνές του βίντεο, οι εναλλαγές (transitions) των σκηνών είναι πάντα αντιληπτές από τους θεατές. Αν δύο εναλλαγές γίνουν σε σύντομο χρονικό διάστημα, τότε μπορεί να εκνευρίσουν κάποιους θεατές. Ο στόχος είναι να κρατήσουμε τις εναλλαγές των σκηνών όσο πιο μακριά γίνεται μεταξύ τους, μεγιστοποιώντας το μήκος της μικρότερης σκηνής του μοντάζ.

Πρόβλημα

Γράψτε ένα πρόγραμμα που παίρνει στην είσοδο δύο γραμμές. Γραμμή 1: οι στίχοι του τραγουδιού, και Γραμμή 2: το κείμενο ενός επεισοδίου του Top Gear. Στη συνέχεια το πρόγραμμα πρέπει να βρει τον καλύτερο τρόπο για να δημιουργήσει τους στίχους από τις υπο-συμβολοσειρές (substrings) του κειμένου του επεισοδίου, κάνοντας μεγιστοποίηση του μήκους της μικρότερης υπο-συμβολοσειράς (substring). Πρέπει να δίνει στην έξοδο το μήκος της μικρότερης υπο-συμβολοσειράς (substring) σε αυτή την παραλλαγή του τραγουδιού. Αν είναι αδύνατο να δημιουργηθεί το τραγούδι, θα πρέπει να δίνει στην έξοδο τον αριθμό -1.

Είσοδος

Το πρόγραμμα πρέπει να διαβάζει από την κανονική είσοδο δύο γραμμές από χαρακτήρες που περιέχουν μόνο κεφαλαία και μικρά γράμματα του αγγλικού αλφαβήτου.

Έξοδος

Το πρόγραμμα πρέπει να τυπώνει στην κανονική έξοδο ένα ακέραιο αριθμό, που είναι το μήκος της μικρότερης υπο-συμβολοσειράς (substring) ή τον αριθμό -1 αν είναι αδύνατη η δημιουργία του τραγουδιού.

Περιορισμοί

Κάθε γραμμή της εισόδου θα έχει το πολύ 100000 χαρακτήρες.

Παράδειγμα

Είσοδος

```
JusticeAndTrust
CarsAndTrucksAreJustNice
```

Έξοδος

```
3
```

Επεξήγηση

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

Υποπροβλήματα

Έστω ότι το *N* είναι το μήκος της πρώτης συμβολοσειράς και το *M* είναι το μήκος της δεύτερης συμβολοσειράς.

subtask	points	*Το μέγιστο των N, M είναι το πολύ:*	σημείωση
1	11	**10**	
2	17	**2 000**	
3	19	**10 000**	
4	23	**30 000**	Η απάντηση στο πρόβλημα είναι μικρότερη ή ίση του 20
5	30	**100 000**	

RADIO

Ο ποταμός Δούναβης είναι το σύνορο μεταξύ Βουλγαρίας και Ρουμανίας. Κάτι που φυσικά έχετε ήδη παρατηρήσει.

Οι άνθρωποι διασχίζουν τον ποταμό με βάρκες. Για την πρόληψη ατυχημάτων, η υπηρεσία διάσωσης έχει κτίσει **N** πύργους ασφαλείας στην Βουλγάρικη όχθη. Οι θέσεις των πύργων ασφαλείας απέχουν $\mathbf{X_1}$, $\mathbf{X_2}$, …, $\mathbf{X_N}$ μέτρα από το σημείο που ο ποταμός μπαίνει στην Βουλγαρία. Για απλοποίηση, θεωρούμε ότι ο Δούναβης είναι ένα ευθύγραμμο τμήμα και οι πύργοι – σημεία με ακέραιες συντεταγμένες πάνω στο ευθύγραμμο τμήμα.

Κάθε πύργος έχει ένα πομπό (radio transmitter). Ο πομπός στον πύργο *i* έχει ισχύ $\mathbf{P_i}$. Οι πύργοι χρησιμοποιούν τους πομπούς για να επικοινωνούν ο ένας με τον άλλο. Δύο πύργοι *i* και *j* μπορούν να επικοινωνούν αν η απόσταση μεταξύ τους είναι μικρότερη ή ίση από το άθροισμα από τις ισχείς των πομπών τους (δηλ. εάν $|\mathbf{X_i} - \mathbf{X_j}| \leq \mathbf{P_i} + \mathbf{P_j}$).

Για να μειωθεί το κόστος, αποφασίστηκε να σταματήσει η λειτουργία κάποιων πύργων. Το νέο πλάνο λέει ότι μόνο **K** πύργοι πρέπει να παραμείνουν και οι υπόλοιποι πρέπει να πωληθούν. Η πώληση του πύργου *i* φέρνει εισόδημα στην κυβέρνηση $\mathbf{S_i}$ leva (lev, πληθυντικός leva, είναι το νόμισμα της Βουλγαρίας), αλλά ο πύργος δεν μπορεί να χρησιμοποιηθεί πλέον.

Παρ'όλα αυτά, για να είναι το σύστημα λειτουργικό, πρέπει να ισχύει ακόμη ότι: κάθε δύο από τους εναπομείναντες **K** πύργους πρέπει να μπορούν να επικοινωνούν μεταξύ τους απευθείας (όταν μόνο ένας πύργος παραμένει, δεν μας ενδιαφέρει η επικοινωνία). Επειδή αυτό πιθανώς είναι αδύνατο με την υφιστάμενη ισχύ κάθε πύργου, οι πύργοι μπορούν να αναβαθμιστούν. Η αναβάθμιση ενός πύργου κοστίζει ένα lev για κάθε μονάδα που αυξάνεται η ισχύς.

Πρόβλημα

Η δεσποινίς Έλλη ξεκίνησε πρόσφατα εργασία στο Υπουργείο Οικονομικών. Τώρα θέλει να προτείνει το πιο οικονομικό σχέδιο για την πώληση **N** − **K** πύργων και την αναβάθμιση των υπόλοιπων, ούτως ώστε κάθε ζεύγος από τους εναπομείναντες πύργους να μπορεί να επικοινωνεί απευθείας. Αποφάσισες να βοηθήσεις το κορίτσι γράφοντας ένα πρόγραμμα το οποίο βρίσκει το μικρότερο κόστος (ή το μεγαλύτερο κέρδος, αν η πώληση των πύργων φέρνει περισσότερα λεφτά από αυτά που ξοδεύονται για την αναβάθμιση τους) για το πρόβλημα.

Είσοδος

Στην κανονική είσοδο, στην πρώτη γραμμή δίνονται δύο ακεραίοι αριθμοί **N** και **K** – ο αρχικός και τελικός αριθμός πύργων, αντίστοιχα. Σε κάθε μια από τις επόμενες N γραμμές δίνονται τρεις ακέραιοι αριθμοί $\mathbf{X_i}$, $\mathbf{P_i}$, $\mathbf{S_i}$ – η θέση του πύργου, η αρχική ισχύς του και η τιμή πώλησης του. Οι πύργοι δίνονται σε αύξουσα σειρά της θέσης τους $\mathbf{X_i}$. Δεν υπάρχουν πύργοι με τις ίδιες συντεταγμένες $\mathbf{X_i}$.

Έξοδος

Σε μια γραμμή της κανονικής εξόδου, το πρόγραμμα πρέπει να τυπώνει ένα ακέραιο αριθμό – το ελάχιστο κόστος (ή μέγιστο κέρδος) για την υλοποίηση του προβλήματος. Σε περίπτωση κέρδους, ο αριθμός πρέπει να είναι αρνητικός.

Περιορισμοί

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

Υποπροβλήματα

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

Παραδείγματα

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

Επεξήγηση

Στο πρώτο παράδειγμα, μια βέλτιστη απάντηση είναι να κρατήσει τους πύργους {1, 3, 4}. Στη συνέχεια πρέπει να αυξηθεί η ισχύς του πύργου 1 κατά 17 και του πύργου 4 κατά 31, πληρώνοντας έτσι 17 + 31 = 48 leva. Η πώληση των υπόλοιπων πύργων (2 και 5) θα αποφέρει 4 + 2 = 6 leva. Το συνολικό κόστος είναι 48 – 6 = 42 leva.

Στο δεύτερο παράδειγμα μπορούμε να επιλέξουμε να κρατήσουμε τους πύργους {2, 3, 6, 7, 9}. Στη συνέχεια, για να μπορούν οι πύργοι να επικοινωνήσουν μεταξύ τους, πρέπει να αυξήσουμε την ισχύ του 7 κατά 2 και την ισχύ του 9 κατά 4, πληρώνοντας έτσι 2 + 4 = 6. Στη συνέχεια μπορούμε να πωλήσουμε τους πύργους {1, 4, 5, 8} για 4 + 6 + 9 + 11 = 30 leva. Το συνολικό κόστος είναι 6 – 30 = −24 leva. Έτσι θα κερδίσουμε 24 leva.

TILING

Ας θεωρήσουμε τους θετικούς ακεραίους *k, l,* και *n*. Οι διαστάσεις του πατώματος ενός ορθογώνιου δωμάτιου είναι $k{\cdot}n \times l{\cdot}n$ μονάδες (το σύμβολο "κεντραρισμένη τελεία" (·) σημαίνει πολλαπλασιασμός ακεραίων). Το πάτωμα έχει πλακοστρωθεί με $k{\cdot}l{\cdot}n$ ορθογώνια πλακίδια μεγέθους $1 \times n$. Κανένα πλακίδιο δεν έχει κοπεί κατά τη διάρκεια της πλακόστρωσης. Μπορούμε να θεωρήσουμε το πάτωμα σαν πλέγμα με $k{\cdot}n$ στήλες και $l{\cdot}n$ γραμμές.

Ένα ψυχικά διαταραγμένο ρομπότ έχει κλειδωθεί στο δωμάτιο και απειλεί να το ανατινάξει αν δεν μαντέψετε τον ακριβή τρόπο της πλακόστρωσης του πατώματος: ποιά είναι η θέση κάθε πλακιδίου στην κάλυψη. Ευτυχώς, το ρομπότ δέχεται να σας δώσει κάποιες πληροφορίες, και με βάση αυτές μπορείτε να προσπαθήσετε να ανακαλύψετε την ακριβή πλακόστρωση. Μπορείτε να ρωτήσετε το ρομπότ ένα σύνολο από *q* ερωτήματα του τύπου *(x y)*, όπου τα $1 \le x \le k.n$ και $1 \le y \le l.n$ είναι αριθμοί, αντίστοιχα, μιας στήλης και μιας γραμμής (η μέτρηση ξεκινάει στο 1 από την «πάνω αριστερή γωνία» – ένα κελί στην πρώτη στήλη και πρώτη γραμμή).

Θα πάρετε πίσω ένα σύνολο από *q* ***σωστές*** απαντήσεις για το ποιό ακριβώς πλακίδιο καλύπτει κάθε ένα από τα ερωτηθέντα κελιά: θα λάβετε τις διαστάσεις της πάνω αριστερά γωνίας («της αρχής») του πλακιδίου και τον τρόπο που είναι τοποθετημένο (την «κατεύθυνσή» του) – «οριζόντια» (κατά μήκος μίας γραμμής) ή «κατακόρυφα» (σε μία στήλη).

Πρόβλημα

Γράψτε μία συνάρτηση `tiling()`, η οποία θα μεταγλωττιστεί με ένα πρόγραμμα της επιτροπής και θα ακολουθεί ένα μικρό διάλογο με το ρομπότ για να ανακατασκευάσει την πλακόστρωση από τις ληφθείσες πληροφορίες.

Ίσως σχεδιάζετε να ρωτήσετε μία ερώτηση για κάθε κελί, και voilà! Παρ' όλα αυτά, το πλήθος *q* των ερωτήσεων που θα θέσετε θα πρέπει να είναι όσο το δυνατόν μικρότερο, ή αλλιώς κινδυνεύετε να ενοχλήσετε το ρομπότ, και παρ' όλο που το πρόγραμμά σας έχει μαντέψει τη σωστή πλακόστρωση, μπορεί να συμβεί μία έκρηξη (αυτό σημαίνει – μηδέν για το test: δείτε τους κανόνες βαθμολόγησης).

Λεπτομέρειες υλοποίησης

Θα πρέπει να υποβάλετε στο σύστημα του grader ένα αρχείο **tiling.cpp**, το οποίο περιέχει τη συνάρτηση `tiling()`. Το αρχείο σας μπορεί να περιλαμβάνει οποιοδήποτε αναγκαίο κώδικα και ονόματα, αλλά δεν πρέπει να χρησιμοποιεί σαν global κανένα από τα παρακάτω δεσμευμένα ονόματα, ούτε και το όνομα `main`.

Στην αρχή του αρχείο θα πρέπει να υπάρχει μία γραμμή με την εντολή προεπεξεργαστή

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

Το πρόγραμμα της επιτροπής ορίζει και υλοποιεί τα παρακάτω στοιχεία:

```
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);
```

Περιγραφή των ορισμένων συναρτήσεων για την επικοινωνία με το ρομπότ:

1. Η συνάρτηση **`getArea`** θα επιστρέφει τις παραμέτρους *k*, *l* και *n* του δωματίου (το πλήθος των στηλών θα είναι $k \cdot n$, και το πλήθος των γραμμών θα είναι $l \cdot n$).
2. Μπορείτε να καλέσετε **μία φορά** τη συνάρτηση **`getData`** με το παραπάνω πρότυπο. Οι παράμετροι έχουν την παρακάτω σημασία:
   - *q* είναι το πλήθος των ερωτήσεων;
   - ο πίνακας εισόδου/εξόδου των εγγραφών `quest` περιλαμβάνει:
   - πριν την κλήση – τις ίδιες τις ερωτήσεις (*x* και *y* είναι αντίστοιχα η στήλη και η γραμμή του ερωτώμενου κελιού). Η τιμή του member *d* δεν παίζει ρόλο εδώ
   - μετά την κλήση – τις απαντήσεις του ρομπότ, δηλαδή: *x* και *y*, που είναι αντίστοιχα η στήλη και η γραμμή της αρχής του πλακιδίου, το οποίο καλύπτει το αντίστοιχο ερωτώμενο κελί, και το member *d,* που έχει ως τιμή ένα από τα δύο παρακάτω σύμβολα: `h`, το οποίο σημαίνει οριζόντια κατεύθυνση, ή `v`, το οποίο σημαίνει κατακόρυφη κατεύθυνση. Αν η συνάρτηση επιστρέψει **`false`**, αυτό σημαίνει ότι είτε υπάρχουν λανθασμένα δεδομένα στον πίνακα εισόδου/εξόδου (για παράδειγμα, εκτός ορίων), είτε η συνάρτηση έχει ήδη κληθεί. Σε αυτή την περίπτωση ο πίνακας θα περιλαμβάνει μηδενικά σαν τιμές εξόδου.
3. Πριν από την έξοδο, η `tiling()` πρέπει να καλεί την ορισμένη συνάρτηση **`setResult`**, όπου ο πίνακας συμβόλων `res` περιγράφει την πλακόστρωση. Αυτά είναι $k \cdot l \cdot n$ σύμβολα (χωρίς διαχωριστικά), κάθε ένα από τα οποία είναι `h` ή `v`. Ακολουθεί ο αλγόριθμος για τη δημιουργία του `res`:

- Ξεκινάμε χωρίς σύμβολα στο `res`;
- Θεωρούμε ότι περπατάμε στο πάτωμα (το πλέγμα) πηγαίνοντας σε γραμμές από την πρώτη στην $l \cdot n$ -οστή, και σε κάθε γραμμή από την πρώτη στήλη στην $k \cdot n$-οστή. Αν πατήσετε στην πάνω αριστερά γωνία ενός πλακιδίου, προσθέτετε ένα σύμβολο στο `res`: `h` αν η κατεύθυνση του πλακιδίου είναι οριζόντια, ή `v` αν είναι κατακόρυφη. Κελιά τα οποία δεν αποτελούν πάνω αριστερές γωνίες (αρχές) απλά παρακάμπτονται.

Περιορισμοί

Σε 20% των tests, $1 \le k, l \le 10$.

Σε 30% των tests, $n = 2$.

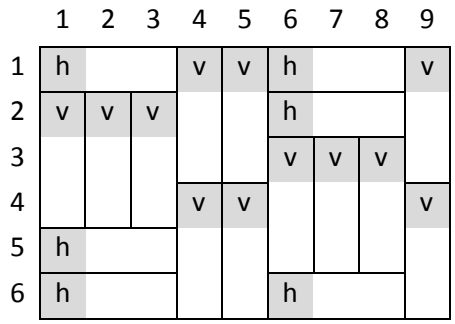
$1 \le k, l \le 57$, $2 \le n \le 10$.

Βαθμολόγηση

Αν η περιγραφή ενός καταχωρημένου πλακιδίου λείπει ή είναι λάθος, δεν δίνονται μονάδες στο test. Μία σωστή περιγραφή προσδίδει μονάδες με βάση το πόσο κοντά είναι το πλήθος των ερωτημάτων q που κάνατε, στο θεωρητικά απαραίτητο πλήθος ερωτημάτων. Πιο συγκεκριμένα, αν το θεωρητικό ελάχιστο είναι Q ερωτήματα και το παράδειγμα του test είναι σχεδιασμένο για P μονάδες, σε μία σωστή περιγραφή θα δοθούν $\min(P, P(Q/q)^{30})$ μονάδες. Το τελικό άθροισμα στρογγυλοποιείται στον κοντινότερο ακέραιο.

Παράδειγμα

Αυτό το παράδειγμα αναφέρεται στην πλακόστρωση του σχήματος. Μπορείτε να δείτε με γκρι χρώμα τις αρχές των πλακιδίων, και ένα γράμμα σε κάθε αρχή, που δηλώνει την κατεύθυνση του πλακιδίου.



Αν, για παράδειγμα, έχετε ορίσει τις μεταβλητές

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

μία κλήση στη συνάρτηση

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

θα θέσει τις τιμές αυτών των μεταβλητών αντίστοιχα σε `k=3, l=2` και `n=3.`

Σαν επεξήγηση για την υποβολή ενός συνόλου ερωτημάτων, κοιτάξτε το παρακάτω απόσπασμα:

```
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)) {//ερμηνεία επιστρεφόμενων δεδομένων}
else {//υπάρχει σφάλμα ή αυτή δεν είναι η πρώτη κλήση}
```

Στο εξεταζόμενο παράδειγμα μετά την πρώτη κλήση της **`getData`**, αφού τα δεδομένα πριν την κλήση είναι μέσα στα όρια του πλέγματος, ο πίνακας `data` θα μοιάζει ως εξής:

```
{{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'}}
```

Το αποτέλεσμα πρέπει να καταχωρηθεί καλώντας την `setResult`. Σε αυτό το παράδειγμα, ας πούμε:

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

Σχόλια για το παράδειγμα

Εννιά σωστά ερωτήματα υποβάλλονται (q=9). Οι επιστρεφόμενες εννιά σωστές απαντήσεις επιτρέπουν τον καθορισμό της πλακόστρωσης, η οποία καταχωρείται σύμφωνα με τους κανόνες. Το πλήθος των υποβληθέντων ερωτημάτων ξεπερνά το θεωρητικό ελάχιστο που είναι απαραίτητο για αυτό το σχηματισμό, το οποίο είναι Q=6. Οπότε σε μια λύση σαν αυτή θα προσδίδονταν ένα μέρος των παρεχόμενων P μονάδων, δηλαδή $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. Όπως μπορείτε να μαντέψετε, μία απάντηση σαν αυτή, αν και σωστή, πρακτικά θα είναι άχρηστη.

Τοπικός έλεγχος

Για να μπορέσετε να ελέγξετε τη συνάρτησή σας *tiling* στον τοπικό σας υπολογιστή, θα λάβετε τα αρχεία *Lgrader.cpp* και *tiling.h*. Μεταγλωττίστε το *Lgrader* μαζί με το αρχείο σας **tiling.cpp** και θα λάβετε ένα πρόγραμμα το οποίο μπορείτε να χρησιμοποιήσετε για να ελέγξετε τη συνάρτησή σας.

Το *Lgrader* διαβάζει το standard input στην παρακάτω μορφή:

- Η γραμμή 1 περιέχει τρεις ακεραίους: k, l και n.
- Ακολουθεί ένας πίνακας ακεραίων. Ο ακέραιος σε κάθε κελί συμβολίζει τον αριθμό ενός πλακιδίου. Ο πίνακας αποτελείται από $l \cdot n$ γραμμές. Κάθε γραμμή περιέχει $k \cdot n$ ακεραίους.

Παράδειγμα για τοπικό έλεγχο (αντιστοιχεί στην παραπάνω εικόνα).

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