

CLARKSON

"Jeremy Clarkson Beatbox" is a popular YouTube video montage of the former Top Gear host performing beatbox. The video consists of a series of scenes from the show, stitched together to produce an entertaining musical performance. The video was published on YouTube in 2009 and it has been viewed almost three million times since then.

This year Jeremy Clarkson departed from the BBC, following a disciplinary ruling. To commemorate the iconic show, we want to produce a sequel to the popular YouTube montage, using scenes from more recent episodes. The lyrics of the new video are already chosen by the fans of the show, but making the montage is not an easy thing.

The problem is that, no matter how skillfully the separate pieces of the video are stitched together, the transitions are always noticeable to the viewers. If two transitions occur within a short period of time, it can even be irritating for some viewers. Therefore, the goal is to keep the transitions as far apart as possible by maximizing the length of the shortest piece in the montage.

Task

Write a program that takes two lines of input: the lyrics of the song on the first line, and the script of an episode of Top Gear on the second line. The program should then find the best way to construct the lyrics out of substrings of the script by maximizing the length of the shortest substring. It should output the length of the shortest substring in that optimal variant of the song. If it's impossible to construct the song out of the given episode, it should output -1 .

Input

Your program should read from the standard input two lines of text, containing only English uppercase and lowercase letters.

Output

Your program has to write to the standard output one integer, which is equal to the length of the shortest substring in the optimal variant of the song, or -1 if construction is impossible.

Constraints

Each line in the input will be at most 100000 characters long.

Example

Input

```
JusticeAndTrust  
CarsAndTrucksAreJustNice
```

Output

```
3
```

Explanation

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

Subtasks

Let us denote the length of the first string by N , and the length of the second string by M .

subtask	points	<i>Max N, M is less than or equal to</i>	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

The border between Bulgaria and Romania is the Danube River. In fact, you have probably already seen it – it's hard to miss.

Since it is possible to cross the river by boat, people in the past and now do it. In order to prevent accidents, the Rescue Service built N security towers on the Bulgarian bank. Their positions are distanced X_1, X_2, \dots, X_N meters downstream from the point where the river enters Bulgaria. For simplicity we will consider the Danube to be a line segment and the towers – points with integer coordinates on it.

There is a radio transmitter in each tower. The transmitter in tower i has power P_i . Using these transmitters the towers might be able to communicate between one-another. Two towers i and j can communicate if the distance between them is less than or equal to the sum of their transmitters' powers (in other words: if $|X_i - X_j| \leq P_i + P_j$).

In order to reduce costs, a decision was made to decommission some of the towers. The new plan states that only K towers must remain and the rest will be sold. Selling tower i brings the government S_i leva (lev, plural leva, is the Bulgarian currency), but the tower can no longer be used.

However, to have a well-functioning system, a new requirement was introduced: each two of the remaining K towers must be able to communicate each other directly (when only one tower remains, we do not mind communications). Since this is potentially impossible with their current powers, they can be upgraded. Upgrading any tower increases its power for the price of one lev per one unit power.

Task

Ms. Elly recently started a summer internship at the Ministry of Finance. Now she wants to shine by proposing the most cost-efficient plan for selling $N - K$ towers and upgrading the rest, so that each pair of the remaining ones can communicate directly between one-another. You decide to help the girl by writing a program which finds the minimal price (or maximal gain, if the sale of towers brings more money than are being spent on upgrading) for the task.

Input

On the first line of the standard input, two integers N and K are given – the number of towers in the beginning and at the end, respectively. On each of the following N lines are given three integers X_i , P_i , S_i – the position of the respective tower, its initial power, and the price at which it can be sold. The towers are given in ascending order of their positions X_i . No two towers have equal coordinates X_i .

Output

On a single line of the standard output, your program should print out one integer – the minimal price (or maximal gain) to do the task. In case of a gain, the printed number should be negative.

Constraints

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

Subtasks

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

Examples

Input	Input
5 3	9 5
4 63 3	5 8 4
13 2 4	10 10 7
87 3 9	11 9 7
121 6 15	13 6 6
159 5 2	19 20 9
	20 2 1
	23 1 3
	26 13 11
	28 4 2

Output	Output
42	-24

Explanation

In the first example, one optimal variant is to keep the towers with indices {1, 3, 4}. This way we must increase the power of tower 1 with 17 and the power of 4 with 31, paying $17 + 31 = 48$ leva. Selling the rest of the towers (2 and 5) we gain $4 + 2 = 6$ leva. The total price is $48 - 6 = 42$ leva.

In the second example we can choose, for example, to keep the towers with indices {2, 3, 6, 7, 9}. In order for the towers to be able to communicate with each other we must increase the power of 7 with 2 and the power of 9 with 4, for a price of $2 + 4 = 6$. With this set of towers we can sell {1, 4, 5, 8} for $4 + 6 + 9 + 11 = 30$ leva. The total "price" is $6 - 30 = -24$ leva. Thus, we have gain of 24 leva.

TILING

Let us consider the positive integers k , l , and n . A rectangular room has floor dimensions of $k \cdot n \times l \cdot n$ units (the sign “centered dot” (\cdot) means multiplication of integers). The floor is tiled with $k \cdot l \cdot n$ rectangular tiles of size $1 \times n$. No tile has been cut during the tiling. We can consider the floor as a grid with $k \cdot n$ columns and $l \cdot n$ rows.

A mentally disturbed robot has locked himself in the room and threatens to blow it up if you don't guess the exact way of floor tiling: what is the position of every tile in the covering. Fortunately, the robot is inclined to give you some information, on which basis you can try to reveal the exact tiling. You can ask the robot a set of q questions of the type $(x \ y)$, where $1 \leq x \leq k \cdot n$ and $1 \leq y \leq l \cdot n$ are numbers, respectively of a column and a row (counting starts at 1 from the “top left corner” – a cell in first column and first row).

You will get back a set of q **correct** answers exactly what tile is covering each of the queried cells: the coordinates of the top left corner (“the beginning”) of the tile and the way it is laid (its “direction”) – “horizontally” (along a row) or “vertically” (in one column).

Task

Write a function `tiling()`, which will be compiled with a jury's program and will lead a short dialog with the robot to reconstruct the tiling out of the received information.

You are maybe planning to ask a question for each cell, and voilà! Alas, the number q of questions asked should be as small as possible, otherwise you take the risk to annoy the robot, and although your program has guessed the right tiling, an explosion can occur (that is – a zero for the test: see the grading rules).

Implementation details

You should submit to the grading system a file **tiling.cpp**, which contains the function `tiling()`. Your file may contain any other necessary code and names, but should use as a global neither one of the predefined names below, nor the name `main`.

At the beginning of the file there should exist one line with the preprocessor instruction

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

The jury's program declares and implements the following items:

```
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);
```

Description of the defined functions for communication with the robot:

1. Function **getArea** will return the room parameters k , l and n (the column count will be $k \cdot n$, and the row count will be $l \cdot n$).
2. You can call **once** the function **getData** with the prototype above. Parameters have the following meaning:
 - q is the number of questions;
 - the input/output array of records `quest` contains:
 - before the call – the questions themselves (x and y are respectively the column and the row of the queried cell). The value of the member named d doesn't matter here;
 - after the call – the answers of the robot, namely: x and y are respectively the column and the row of the beginning of the tile, covering the respective asked cell, and the member d has for its value one of these two symbols: h , meaning horizontal direction, or v , denoting vertical direction. If the function returns **false**, that means that either there are incorrect data in the input/output array (say, out of the borders), or the function has already been called. In this case the array will contain zeroes as output values.
3. Before the exit, `tiling()` should call the defined function **setResult**, where the symbol array `res` describes the tiling. These are $k \cdot l \cdot n$ symbols (with no delimiter), each being h or v . Here follows the algorithm for creating `res`:
 - Start with no symbols in `res`;
 - Consider walking the floor (the grid) by rows from the first to the $l \cdot n$ -th, and every row by columns from the first to the $k \cdot n$ -th. If you step on an upper left corner of a tile, you add one symbol to `res`: h for horizontal direction of the tile, or v for vertical direction. Cells which are not upper left corners (beginnings) are being simply jumped over.

Constraints

In 20% of the tests, $1 \leq k, l \leq 10$.

In 30% of the tests, $n = 2$.

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

Grading

If a registered tiling description is missing or wrong, the test is given no points. A correct description is granted points according to the proximity of the asked questions' count q to the theoretically necessary questions' count. More precisely, if the theoretical minimum is Q questions and the test example is designed for P points, a correct description will be given $\min(P, P(Q/q)^{30})$ points. The final sum is rounded to the nearest integer.

Example

This example is made with respect to the tiling in the picture. You can see in grey the beginnings of the tiles, and a letter in each beginning, denoting the direction of the tile.

If, for example, you have defined variables

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

a call of the function

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

will set the values of these variables respectively to $k=3$, $l=2$ and $n=3$.

For an illustration about posing a set of questions, look at the following fragment:

```
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 }
```

In the considered example after the first call of `getData`, as the data before the call are in the border limits, the array `data` will look like this:

	1	2	3	4	5	6	7	8	9
1	h			v	v	h			v
2	v	v	v			h			
3						v	v	v	
4				v	v				v
5	h								
6	h					h			

```
{ {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'} }
```

The result should be registered by calling `setResult`. In this example, say:

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

Example comments

Nine correct questions are posed ($q=9$). The returned nine correct answers allow tiling's determination, which is registered in accordance with the rules. Posed question's count exceeds the theoretical minimum needed for this configuration, which is $Q=6$. So a solution like this would be granted a part of the provided P points, namely $\{P(2/3)^{30}\} \approx \{0.000005 P\}$. As you can guess, an answer like this, although correct, will practically be useless.

Local testing

In order to be able to test your function *tiling* on your local computer, you will get files *Lgrader.cpp* and *tiling.h*. Compile *Lgrader* together with your file **tiling.cpp** and you will receive a program that you can use to test your function.

Lgrader reads the standard input in the following format:

- Line 1 contains three integers: k , l and n .
- It follows a matrix of integers. The integer in each cell of this matrix denotes a tile number. The matrix consists of $l \cdot n$ lines; each line contains $k \cdot n$ integers.

Example for local testing (corresponds to the picture above).

Input	Output
3 2 3	hvvhvwwwvhvvvvvvhhh
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	