Problem A. Consecutive Sums

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	1 second
Time limit (Java):	2 seconds
Time limit (Python):	10 seconds
Memory limit:	64 megabytes
Java Class Name:	`sums`
Maximum Points Available:	100

The many days of continuous riding had proved hard for Sir Arnold. The weight of his armor was tiring him and the monotonous landscape made it seem as if no progress was being made at all.

To make it all worse, that new page of his, Jeremy, would not stop talking. Sir Arnold had given up listening to him three towns back, but this seemed to make no difference.

In an attempt to distract himself from the journey, Sir Arnold was practising his arithmetic. He would choose two positive numbers, S and T and sum all the numbers between them to get N:

$$N = S + (S + 1) + (S + 2) + \ldots (T - 1) + T.$$

After a while, he wondered which numbers N he could obtain this way, and which he couldn't. Your task is to write a program to determine if N can be calculated in this way or not.

Input

The input consists of a single line, containing a single integer, N.

Output

Output consists of two space separated integers on a single line, the first number in the sum, S, and the last number in the sum, T.

If no such sum exists, then the output should be a single line containing the string number -1.

Scoring

In cases worth a minimum of 20 points, $1 \leq N \leq 20$.

In cases worth a minimum of 40 points, $1 \leq N \leq 1\,000\,000$.

In cases worth a minimum of 40 points, $1 \leq N \leq 10^{16}$.

Examples

standard input	standard output
10	1 4
8	-1
15	4 6

Note

In the examples given, $1 + 2 + 3 + 4 = 10$ and $4 + 5 + 6 = 15$. However, there are no ways to write 8 as the sum of consecutive positive integers.

Problem B. Beautiful Words

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	3 seconds
Time limit (Java):	6 seconds
Time limit (Python):	30 seconds
Memory limit:	256 megabytes
Java Class Name:	`words`
Maximum Points Available:	100

Bernard the monk, is transcribing a book.

Unfortunately, Bernard is not a very good monk, and therefore doesn't care if his transcription is accurate. He is also quite an artistic soul, and therefore wants every word he writes to be as beautiful as possible.

He has assigned a beauty to a word as follows: for each pair of adjacent letters x and y, a bonus beauty of $c(x, y)$ is added to the beauty of the word.

Thus, if `c(p,p)=3` and `c(a,r)=2` and `c(h,a)=1`, then "happy" has a beauty of $1 + 3 = 4$ and "harry" has a beauty of $2 + 1 = 3$.

After Bernard has written a page, another monk will look over the page to confirm that it is correct. Bernard knows that he can change up to K letters in a word before the second monk notices and makes him rewrite the whole page.

Given a word, and the bonuses for letter pairs, help Bernard find the most beautiful word he can make, changing at most K letters. It doesn't have to be a "real" word. It just has to look beautiful.

Input

The first line contains a string S, the initial word, and integer K.

The second line contains an integer, N ($0 \leq N \leq 676$), the number of pairs of letter with bonuses. The next N lines contain two letters and an integer each, denoting that the pair of letters (in order given) gives that bonus.

No pair of letters will be mentioned more than once in the data. All values for bonuses will be between -10^6 and 10^6 inclusive.

Output

Output a single integer: the maximum possible beauty Bernard can achieve.

Scoring

We use $|S|$ to denote the length of a string, S.

In cases worth a minimum of 15 points $1 \leq |S| \leq 4$.

In cases worth a minimum of 20 points $1 \leq |S| \leq 200$, $K = |S|$.

In cases worth a minimum of 15 points $1 \leq |S| \leq 50$, $0 \leq K \leq 10$.

In cases worth a minimum of 50 points $1 \leq |S| \leq 1\,000$, $0 \leq K \leq 100$.

Examples

standard input	standard output
beauty 3 4 e a 12 a n 9 n i 10 i e 8	39
abcxyz 3 5 a b -20 b c 5 c d 5 d e 5 e f 5	15

Note

In the first example, the most beautiful word that can be obtained from `beauty` is `beanie` with a beauty value of 39.

In the second example, the most beautiful word can be obtained by changing the first letter to something other than `a` and then changing `x` to `d` and `y` to `e` to give a total beauty value of 15.

Problem C. Lost Cartographers

Time limit (C++):	1 second
Time limit (Java):	2 seconds
Time limit (Python):	10 seconds
Memory limit:	64 megabytes
Java Class Name:	`lost`
Maximum Points Available:	100

As monarch of the Kingdom of a Thousand Marshes (And Two Forests) you have decreed that an updated map of the Kingdom must be made.

A fortnight ago you dispatched two sets of cartographers with instructions to go to the North to map out the bogs and marshes there. Last night though, carrier pigeons arrived with dire news.

It seems that heavy rains have caused flash floods and both parties have lost their horses. Furthermore, winter is at hand, and your cartographers might not survive in the cold. You discern that their best chance for survival is for the two parties to join together and pool their resources.

Unfortunately, their maps were also lost with the flood, and so they have no idea where they are. However, they do send a small map of the region just around them.

In a moment of brilliance, you have decided that, using carrier pigeons, you will lead them to one another.

More precisely, the cartographers are on a grid and can at any time see a small area around them. Many of the squares on the grid (which is of size at most $N \times N$) are passable, but some squares are marsh and your cartographers cannot walk there.

You can communicate with the parties in turn and receive from them a map of the $M \times M$ blocks around them (in all cases, M will be odd). This will show which blocks are passable and which impassable. You may then tell them to move one block to the North, South, East or West.

Each pigeon takes a day to get between you and the cartographers, and so you must act quickly to bring them together as soon as possible. You estimate 10 000 pigeons should be the maximum before both parties perish.

Interaction Protocol

This is an interactive task. You are thus required to compile your code against a stub that will perform the interaction with the grader for you.

You are required to write two functions, `init` and `makeMove`. The functions must have the following signatures

Python
```
def init(N,M)
def makeMove(turn, map)
```

C++
```
void init(int N, int M);
char makeMove(int turn, std::string map[])
```

Java
```
public static void init(int N, int M)
public static char makeMove(int turn, String[] map)
```

When your program is run the function `init` will be called once with the size of the grid, N and the size of the maps returned by the cartographers, M.

Subsequently, the stub will call `makeMove` until either the number of turns runs out, or the two parties are on the same square, or an illegal move is made.

The first parameter to `makeMove` will either be 1 or 2, depending on which group of cartographers' pigeon you have just received. The parameter will alternate with callings. That is, the first time `makeMove` is called, `turn` will be 1. The second time 2, the third time 1 again and so on.

The second parameter to `makeMove` is an array of M strings. Each string is of length M. This array describes the $M \times M$ grid around the cartographers. The cartographers are at the center of the grid.

A `.` character represents a passable square and a `#` character an impassable one.

You must return a single character, namely either `N`, `S`, `E` or `W` denoting the direction in which the party is to move one block.

The grader will then call `makeMove` repeatedly as described above.

Example

In this simple example, the two parties are very close together and meet after each has moved two squares.

Function Call	Function Response
`init(20,7)`	None

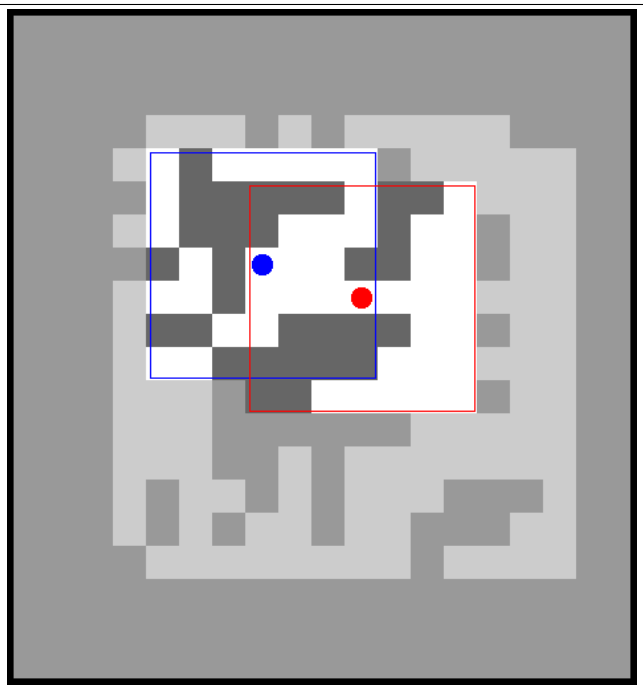
`makeMove(1,{`
`'.#.....',`
`'.#####.',`
`'.###...',`
`'#.#...#',`
`'..#....'}`
`'##..###'}`
`'..#####'}`
	`E`

`makeMove(2,`
`'###.##.',`
`'#...#..',`
`'...##..',`
`'.......',`
`'.####..'}`
`'####...'}`
`'##.....'}`
	`E`

`makeMove(1,`
`'#.....#',`
`'#####.#',`
`'###...#',`
`'.#...##',`
`'.#.....'}`
`'#..####'}`
`'.#####.'}`
	`E`

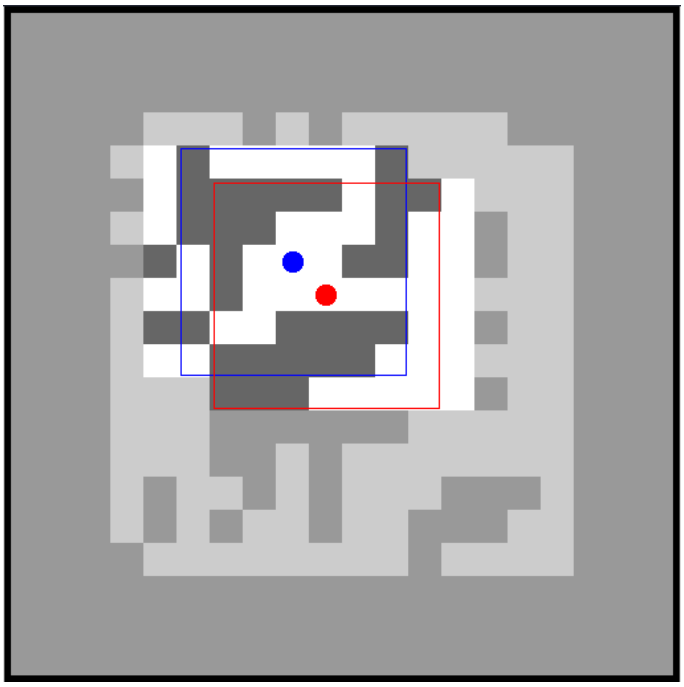
`makeMove(2,`
`'####.##',`
`'##...#.',`
`'#...##.',`
`'#......',`
`'..####.'}`
`'#####..'}`
`'###....'}`
	`N`

The starting configuration looks as follows



The dark squares are impassable and the light squares are passable. The "greyed out" squares are those that haven't been seen by either party yet. The parties are represented by red and blue dots, and their field of view is represented by the thin rectangles.

After two moves (ie one move each, the board looks like this:



After another two moves, the two parties are on the same square and the program exits.

Scoring

In test cases worth at least 25 points, the board will have no obstructions, except for the entire border

In test cases worth at least 25, the board will have at most 25% of its squares (except for the border) impassable.

For the remainder of the test cases, no further restrictions apply.

It will always be possible for the parties to meet each other eventually.

If an illegal move is made, or the parties are not on the same square after 10 000 moves in total, then the score for the subtask will be 0.

Otherwise the score will be calculated as follows:

$$S = \max\left(0, 1 - \frac{t}{N^2}\right)$$

where t is the number of doves sent before the parties are on the same block, and N is as described above.

Note

Testing your program is made easier with the supplied stub code and manager.

To compile and test your program, run the script `run.sh` with the name of an input file and the name of your source code. The script will automatically compile your code against the stub, and run it against the grader. It will output your code's responses to the input.

The format of the input file is as follows. The first line must contain three space separated integers, N, M and the case number. The next N lines must describe the board, and the final two lines specify the coordinates of the two parties, measured from (1,1) in the top right. Some samples will be provided for you.

For example, to test a solution written in java against the sample input, you must run

```
./run.sh primes.java sample.in
```

If your stubs, graders or compilation scripts are corrupted during the contest, you may re-download them from the problem statement on the web interface.