problem	tomo	zmija	autoput

source file	`tomo.pas`
	`tomo.c`
	`tomo.cpp`
		`zmija.pas`
		`zmija.c`
		`zmija.cpp`
			`autoput.pas`
			`autoput.c`
			`autoput.cpp`

input data	stdin		
output data	stdout		
time limit (Athlon 64 3000+)	1 sec	1 sec	2 sec
memory limit (heap)	32 MB		
memory limit (stack)	8 MB		
points	**30**	**40**	**80**
	150		

The most advanced feature of Tom's old calculator is that when he multiplies two numbers and continues to press the button '=', calculator continues to multiply the displayed number with the second of the two entered numbers. For example, if he multiplies 2 and 3, the number 6 will appear on the display. Further pressing the button '=' will result in numbers 18, 54, 162, etc. being displayed.

Tom is playing one interesting game – he wants to know how many times does he need to press the '=' button in order for some given number to appear as suffix of the number displayed on the calculator.

More precisely, Tom enters a number A, presses the button '*', enters the number B and presses the button '='. After that, product of the numbers A and B appears on the display. If the number C is not a suffix of the number displayed, Tom continues to press the button '=', until a number whose suffix is C is displayed.

We say that number X is **suffix** of number Y if the decimal representation of number Y ends with the decimal representation of number X (for example, 46 is suffix of 1246, but 70 is not suffix of 4701). We assume that the calculator can handle numbers **with arbitrary many digits** and that it **can display** all those digits.

Write a program that will, given numbers A, B and C, calculate the total number of times the button '=' will be pressed.

input data

First and only line of input contains integers A, B and C, $0 \leq A, C \leq 100{,}000$, $0 \leq B \leq 1000$. There will be no leading zeros in the representation of number C.

output data

First and only line of output should contain the number of times the button '=' will be pressed, or the word '**NIKAD**' if there is no solution.

examples

input	**input**	**input**
1 2 3	2 3 4	5 3 215
output	**output**	**output**
NIKAD	3	5

One of the most popular DOS games is '**Dummy**'. The snake is crawling through the board and eating apples that increase its length. The game **ends** when the snake **bumps into itself or into the wall**.

Game board consists of NxN squares arranged in N rows and N columns, and some squares contain apples. Around the board there is a wall. At the beginning of the game, the snake is located in the **top-left corner**, its length is equal to **1** and its head is **directed towards right**.

Snake is crawling by changing its position **during each second** according to the following rules:

· first, snake extends its length by spreading to the next square in front of the head (i.e. in the direction of the head),

· if there is an apple on that square, tail of the snake does not move (hence, its length is increased by 1 in this step),

· if there is no apple, last square of the tail is erased (hence, its length stays unchanged)

Positions of the apples and movements of the snake are given. Write a program that will calculate the **number of seconds** until the game ends.

input data

First line of input contains an integer N, $2 \leq N \leq 100$.

Following line contains an integer K, $0 \leq K \leq 100$, the number of apples on the board.

Following K lines contain coordinates of apples on the board. First number denotes the row and second number denotes the column of the board where the apple is situated. There will be no apple at the top-left corner of the board.

Following line contains an integer number L, $1 \leq L \leq 100$, the number of times the snake makes a turn.

Each of the following L lines contains the description of one turn. A single turn is described by a number X (positive integer **less than or equal to 10,000**) and a character C. This means that the snake **rotates its head** 90 degrees left (if C is '**L**') or right (if C is '**D**') at the **end of the Xth second**

output data

First and only line of output should contain number of seconds from the problem statement.

examples

input

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

output

```
9
```

input

```
10
4
1 2
1 3
1 4
1 5
4
8 D
10 D
11 D
13 L
```

output

```
21
```

input

```
10
5
1 5
1 3
1 2
1 6
1 7
4
8 D
10 D
11 D
13 L
```

output

```
13
```

Imagine one simple road in a coordinate system. Road is going from left to right, following the configuration of the land and within one square it can:

a) stay on the same height

b) go down or up by one square

Car is driving on the road from left to right. The time needed to travel through a single square is either **A seconds** for the case a), or **B seconds** for the case b).

However, we can build a **tunnel** under some mountain or a **viaduct** above some valley. These objects have to be **horizontal**, and the time needed for a car to travel through a single square on a tunnel or viaduct is **C seconds**.

Write a program that will, for a given configuration of land, calculate the **minimal time** for a car to travel the whole road with optimal construction of tunnels and viaducts. Total number of objects built **must not be greater than the given number K**.

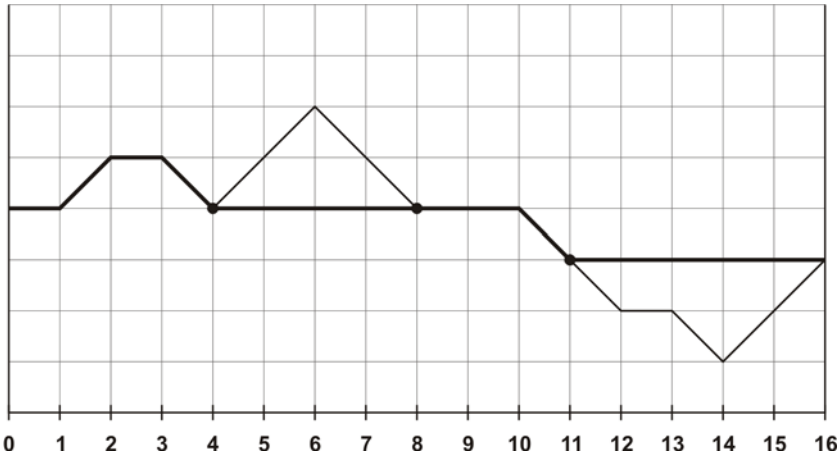

Figure above corresponds to the third example. Original road is denoted by the thin line, and the optimal path is denoted by the thick line. Because the number of objects is restricted to 2, we could not build a tunnel under the first mountain.

input data

First line of input contains three integers A, B and C, $1 \leq A,B,C \leq 100$.

Second line of input contains two integers N and K, $1 \leq N \leq 100{,}000$, $1 \leq K \leq 300$.

Third line of input contains a sequence of N characters that describes the configuration of the land, from left to right. Each character in the sequence is one of the following:

'D' in next square land is going DOWN

'R' in next square land is staying on the SAME HEIGHT

'G' in next square land is going UP

output data

First and only line of output should contain the minimal time from the problem statement.

examples

input

```
3 2 1
9 1
GGDGGDDRR
```

output

```
16
```

input

```
3 5 4
10 10
RGDRDRRRRG
```

output

```
36
```

input

```
10 20 15
16 2
RGRDGGDDRRDDRDGG
```

output

```
235
```