TASK	OKUPLJANJE	USPON	RAZINE	ABECEDA	STEP	VODA

source code	`okupljanje.pas`
	`okupljanje.c`
	`okupljanje.cpp`
		`uspon.pas`
		`uspon.c`
		`uspon.cpp`
			`razine.pas`
			`razine.c`
			`razine.cpp`
				`abeceda.pas`
				`abeceda.c`
				`abeceda.cpp`
					`step.pas`
					`step.c`
					`step.cpp`
						`voda.pas`
						`voda.c`
						`voda.cpp`

input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	1 second	1 second	5 second
memory limit	32 MB	32 MB	32 MB	32 MB	32 MB	128 MB
point value	**30**	**50**	**70**	**100**	**120**	**130**
	500					

The day after a giant party everybody wants to know a couple of things - who was at the party and how many people were there? Since parties are usually pretty big, nobody actually knows the correct number of people who were there. Your friend Krešo was on a party last Saturday and he knows how many people there were per 1 m^2.

While reading 5 newspaper articles about that party, you have written down 5 numbers, specifying how many people were present at the party according to each of the articles. You believe Krešo's information and you'd like to know how much each of the articles was wrong.

INPUT

The first line of input contains two positive integers, **Lj** ($1 \leq$ **Lj** ≤ 10), the number of people per m^2, and **P** ($1 \leq$ **P** ≤ 1000), the area of the room the party was held in.

The second line of input contains 5 positive integers less than 10^6, the number of people present at the party according to each of the articles.

OUTPUT

The first and only line of output must contain 5 numbers, the difference between the number of people written in an article and Krešo's (correct) number.

SAMPLE TESTS

input	**input**
1 10 10 10 10 10 10	5 20 99 101 1000 0 97
output	**output**
0 0 0 0 0	-1 1 900 -100 -3

Tomislav has recently discovered that he's completely out of shape. He actually becomes tired while walking down the stairs! One morning he woke up and decided to come in good shape. His favourite sport is cycling, so he decided to ride a tour on the local hills.

The route he is taking is described as a sequence of **N** numbers which represent the height of the road at evenly spaced points of the route, from the beginning to the end of it. Tomislav is interested in the **largest segment of the route which goes up the hill** he has to ride, according to the information he has. Let's call such a segment a **'climb'**. Tomislav is too tired to bother about details, so he will only take into account the height difference of a climb, not its length.

A climb is more strictly defined as a consecutive increasing subsequence of at least two numbers describing the road. The size of the climb is the difference between the last and first number in the subsequence.

For example, let's consider a route described by the following sequence of heights: `12 3 5 7 10 6 1 11`. Underlined numbers represent two different climbs. The size of the first climb is 7. The second climb is larger, with size 10. Points with heights 12 and 6 are not parts of any climb.

Help Tomislav and calculate the largest climb!

INPUT

The first line of input contains a positive integer **N** ($1 \leq \mathbf{N} \leq 1000$), the number of measured points on the route.

The second line of input contains **N** positive integers $\mathbf{P}_i$ ($1 \leq \mathbf{P}_i \leq 1000$), the heights of measured points on the route.

OUTPUT

The first and only line of output should contain the size of the largest climb. If the route in the input does not contain any climbs, output 0.

SAMPLE TESTS

input	**input**	**input**
5 1 2 1 4 6	8 12 20 1 3 4 4 11 1	6 10 8 8 6 4 3
output	**output**	**output**
5	8	0

Second sample description: climbs are 12-20, 1-3-4, and 4-11. 1-3-4-4-11 is not a climb because sequence of numbers describing a climb has to be strictly increasing.

Mirko has developed his own video game. The game has **N** levels and each successfully completed level is worth a certain number of points, which add up to the player's total score on an online rank list of all players. Mirko has ordered his levels **by difficulty** from the easiest to the most difficult, but he has made a mistake and made some difficult levels worth less points than some of the easier ones.

To overcome this problem, Mirko has decided to **reduce** the number of points for certain levels with the goal of making the point sequence **strictly increasing** (so in the end easier levels are worth less points than the difficult ones).

Help Mirko fix his video game in such a way that the **total number of points reduced is minimal**. Final points have to be positive. You can assume that a solution exists for each test case.

INPUT

The first line of input contains one positive integer N ($1 \leq \mathbf{N} \leq 100$), the number of levels.

The next **N** lines contain positive integers less than 20 000, the number of points that Mirko has associated with each level, from the first to the last level.

OUTPUT

The first and and only line of output should contain one number - the minimum total number of points Mirko has to subtract to fulfill requirements given in the task statement above.

SAMPLE TESTS

input	**input**
3 5 5 5 **output** 3	4 5 3 7 5 **output** 6

A list of words written in some unknown alphabet was found. It is known, however, that these words are in alphabetic order.

Write a program that will find the unique alphabetic ordering of used letters, or determine that no such ordering exists or that there is more than one possible solution.

INPUT

The first line of input contains a positive integer **N** (**N** ≤ 100), the number of words.

The following **N** lines contain the list of words found, one word per line. Each word consists of at most 10 lowercase letters.

OUTPUT

The first and only line of output should contain all letters in alphabetic order. If no such ordering exists, output '!'. If there is more than one solution, output '?'.

SAMPLE TESTS

input	**input**	**input**
5 ula uka klua kula al **output** luka	4 jaja baba baja beba **output** !	3 marko darko zarko **output** ?

Mirko and Slavko started taking tap dance lessons. This dance consists mostly of tapping the floor with a special kind of shoe. Since Mirko and Slavko are fast learners, they decided to come up with their own choreography.

Tap dance choreography can be described as a sequence consisting of two letters, 'L' and 'R'. 'L' means that you should tap the floor with your left foot, and 'R' with your right foot. Mirko realised that the most exciting parts of tap dancing are the ones in which you don't use the same leg twice in a row. He defined the **value** of a choreography as the longest subsequence of consecutive elements that doesn't contain two consecutive 'L's or 'R's.

As we all know, designing a choreography can be very challenging, with lots of small changes until it's done. For every alteration that Slavko does, he would like to know the current choreography value. One alteration is changing one 'L' to 'R', and vice versa.

Before any alterations are made, the choreography consists only of letters 'L'.

INPUT

The first line of input contains two integers, the choreography length **N** ($1 \leq \mathbf{N} \leq 200\,000$), and the number of alterations **Q** ($1 \leq \mathbf{Q} \leq 200\,000$).

Each of the next **Q** lines contains an integer specifying the position that Mirko and Slavko are altering, in order of alteration.

OUTPUT

The output must contain **Q** integers, one per line - the current values of the choreography after each alteration.

SAMPLE TESTS

input

```
6 2
2
4
```

output

```
3
5
```

input

```
6 5
4
1
1
2
6
```

output

```
3
3
3
5
6
```

First sample description: choreographies are: LLLLLL → L**R**LLLL → LRL**R**LL

Mirko, the mad plumber, was hired to construct a water supply network between two locations in a city. The city map can be represented as a **R** × **S** grid. Some cells are **not suitable** for placing water pipes. Locations Mirko needs to connect are placed **directly above** the top left cell of the grid, and **directly below** the bottom right cell.

Each suitable cell Mirko can either **leave empty** or use it for placing **one of the following 6 pipe types**:

Find the number of ways that pipes can be placed to connect the two locations with a continuous pipe (water must not be spilled). All placed pipe parts **must be in use**.

Output the solution **modulo 10007**.

INPUT

The first line of input contains the integers **R** and **S** ($2 \leq \mathbf{R}, \mathbf{S} \leq 10$), the number of rows and columns of the city grid, respectively. Each of the next **R** lines contains exactly **S** characters: '.' if the cell is suitable for placing pipes, and '#' if not.

OUTPUT

The first and only line of output must contain the required number of ways modulo 10007..

SAMPLE TESTS

input	input
2 3 ... .#.	3 3 ... ...
output	**output**
1	12

First sample description: this is the only possible solution: