COCI 2016/2017

Round #3, November 26th, 2016

Tasks

Task	Time limit	Memory limit	Score
Imena	1 s	32 MB	50
Pohlepko	1 s	64 MB	80
Kronican	2 s	32 MB	100
Kvalitetni	1 s	64 MB	120
Zoltan	1 s	32 MB	140
Meksikanac	1.5 s	256 MB	160
Total			650

Little Mirko likes to type and often gets bored during class, which is why his teacher assigned him a task.

Mirko must retype **a book** that contains *N* space-separated sentences. In this book, **a sentence** is an array of one or more space-separated words, where only the last word's character is a punctuation mark ('.', '?' or '!'). The rest of the words do not contain punctuation marks.

Words are **arrays of characters**, either lowercase or uppercase letters of the English alphabet, or digits or, exceptionally, a punctuation mark at the end of the last word in the sentence.

Although he likes typing sentences, Mirko doesn't like typing names. **A name** is a word that starts with an uppercase letter of the English alphabet, whereas the rest of the characters are lowercase letters of the English alphabet, except the last character, which can be the punctuation mark. Before he decides to retype the whole thing, Mirko wants to know **how many names there are in each sentence of the book**. Write a programme to help him!

INPUT

The first line of input contains the integer N ($1 \le N \le 5$), the number from the task.
The second line contains N sentences from the book. The total number of characters in the book will not exceed 1000.

OUTPUT

You must output N lines. The i^{th} line contains the number of names in the i^{th} sentence.

SCORING

In test cases worth 40 points total, it will hold N = 1.

SAMPLE TESTS

input	input
1	2
Spavas li Mirno del Potro Juan martine?	An4 voli Milovana. Ana nabra par Banana.
output	output
4	1
	2

Clarification of the second test case:

The word “An4” starts with an uppercase letter, but contains a digit and therefore can’t be a name.

Little Greedy got a board for his birthday. The board has *N* rows and *M* columns, and has a **lowercase letter of the English alphabet** in each field. During his birthday party, everyone got bored so they decided to play a simple board game.

The game begins with placing a chip on the **upper left** field labeled with coordinates (1, 1). In each turn, we **must** move the chip one field to the right or down, given the constraint that it remains on the board. The game ends with moving the chip to the **lower right** field of the board labeled with coordinates (*N*, *M*). During the game, we take note of the array of characters we form by moving the chip and therefore constructing a word. The goal of the game is to find the lexicographically smallest word.

The player(s) that will suceed in constructing the lexicographically smallest word get a bag of candy as a prize. Greedy wants to win the candy at any price, so he is asking you to write a programme that will find the lexicographically **smallest** possible word.

Please note: The lexicographic order of words is the one in which the words appear in a dictionary. If we have two words, and the words differ in the first letter, then the smaller word is the one with the letter that comes first in the alphabet.

INPUT

The first line of input contains integers *N* and *M*, separated by space ($1 \le N, M \le 2000$).
The following *N* lines contains *M* lowercase letters of the English alphabet that represent the board.

OUTPUT

You must output the lexicographically smallest word.

SCORING

In test cases worth 40 points total, it will hold that, for each field, the letters located to the right and below will be different.

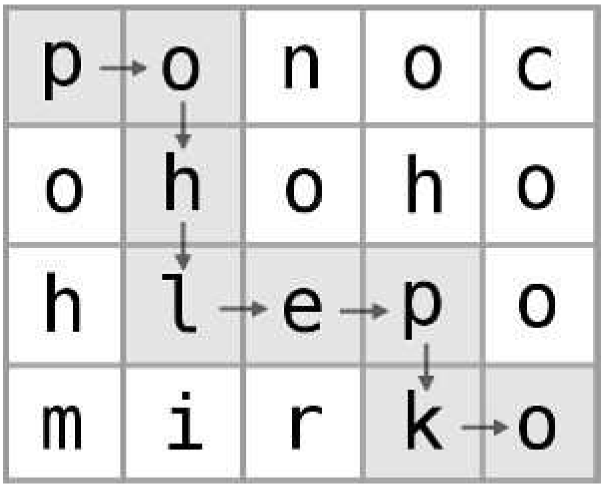
SAMPLE TESTS

input	input	input
4 5 ponoc ohoho hlepo mirko	4 5 bbbbb bbbbb bbabb bbbbb	2 5 qwert yuiop

output	output	output
pohlepko	bbbbabbb	qweiop

Clarification of the first test case:

One way of constructing the smallest word is illustrated in the following image:



Little Mislav owns N glasses of infinite volume, and each glass contains some water. Mislav wants to drink all the water, but he doesn't want to drink from more than K glasses. What Mislav can do with the glasses is pour the total volume of water from one glass to another.

Unfortunately, it matters to Mislav what glass he picks, because not all glasses are equally distant to him. More precisely, the amount of effort it takes to pour water from glass labeled with i to glass labeled j is denoted with C_{ij}.

Help Mislav and find the order of pouring the water from one glass to another such that the sum of effort is minimal.

INPUT

The first line of input contains integers N, K ($1 \le K \le N \le 20$).
The following N lines contains N integers C_{ij} ($0 \le C_{ij} \le 10^5$). The i^{th} row and j^{th} column contains value C_{ij}. It will hold that each C_{ii} is equal to 0.

OUTPUT

Output the minimal effort needed for Mislav to achieve his goal.

SCORING

In test cases worth 40 points total, it will hold $N \le 10$.

SAMPLE TESTS

input	input	input
3 3 0 1 1 1 0 1 1 1 0	3 2 0 1 1 1 0 1 1 1 0	5 2 0 5 4 3 2 7 0 4 4 4 3 3 0 1 2 4 3 1 0 5 4 5 5 5 0
output	**output**	**output**
0	1	5

Clarification of the first test case: Mislav doesn't need to pour water in order to drink from at most 3 glasses.
Clarification of the second test case: Mislav must pour the water from precisely one (doesn't matter which) glass into any other glass in order to be left with only two glasses with water.

Clarification of the third test case: In order for Mislav to achieve the minimal solution of 5, he can pour water from glass 4 to 3 (effort 1), then 3 -> 5 (effort 2), and finally, 1 -> 5 (effort 2). In total, 1 + 2 + 2 = 5 amount of effort.

A quality arithmetic expression consists of brackets, number and operations of multiplication and addition.

A quality arithmetic expression is defined recursively in the following way:

- An expression consisting of only one **positive real** number smaller than or equal to Z_1 is of good quality.
  Such expression is of the following form:
  $$(x)$$
  For example, if $Z_1 = 5$, then (4) is a quality expression.
- If $A_1, A_2, \ldots, A_k$ are quality expressions such that $2 \le k \le K$ and **the sum** of these expressions is at most Z_k, then the following expressions are of good quality:
  $$(A_1+A_2+...+A_k)$$
  $$(A_1*A_2*...*A_k)$$

You are given a quality expression where the numbers are replaced by question marks. Determine the **maximal** possible value that the expression could have had.

INPUT

The first line of input contains integer K ($2 \le K \le 50$).
The second line of input contains integers $Z_1, \ldots, Z_K$, separated by space ($1 \le Z_1, \ldots, Z_K \le 50$).
The third line of input contains one quality arithmetic expression in the described format. Arithmetic expression consists of: '?', '*', '+', '(', ')', and its length is 1 000 000 characters, at most.

OUTPUT

You must output the maximal possible value of the expression.
A solution is considered correct if the absolute or relative deviation from the official solution is less than 10^{-3}.

SAMPLE TESTS

input	input	input
2 10 6 ((?)+(?))	3 2 5 3 (((?)+(?))*(?))	3 2 10 6 ((?)*(?)*(?))
output	output	output
6.00000	6.00000	8.000000000

Clarification of the first test case:
The expression ((3)+(3)) satisfies the conditions, so it is a quality expression, and it is easy to check that 6 is the maximal value.

Clarification of the second test case:
The maximum is achieved for, for instance, the expression (((1)+(2))*(2)).

Clarification of the third test case:
The maximum is achieved for, for instance, the expression ((2)*(2)*(2)).

Marton's friend Cero has an array of N positive integers. In the beginning, Cero writes the first number on the board. Then he writes the second number to the left or to the right of the first number. After that, he writes the third number to the left or to the right of all the numbers written so far, and so on.

Marton asked Cero what the length of the longest possible strictly increasing subsequence (not necessarily of consecutive elements) was.

He also wants to know the number of such longest strictly increasing subsequences. More precisely, if the length of the longest increasing subsequence is M, he wants to know the sum of numbers of strictly increasing subsequences of length M for each possible sequence that Cero can construct. The sequences are different if they are constructed using a different order of moves, and the subsequences in a constructed sequence are different if they differ in at least one position.

Given the fact that the number of such subsequences can be extremely large, Marton will be satisfied with the value of that number modulo $10^9 + 7$.

Cero really doesn't have time at the moment to find out the answers to Marton's questions, so he is asking you to do it for him.

INPUT

The first line of input contains the integer N ($1 \le N \le 2 * 10^5$).
The following line contains N space-separated integers that represent the elements of Cera's array. Each number in the input will be smaller than or equal to 10^9.

OUTPUT

You must output, in a single line, the length of the longest strictly increasing subsequence and the number of strictly increasing subsequences of that length, modulo $10^9 + 7$, respectively.

SCORING

In test cases worth 30% of total points, it will hold $N \le 20$.
In test cases worth 50% of total points, it will hold $N \le 1000$.

SAMPLE TESTS

input 2 1 1 output 1 4	input 4 2 1 3 4 output 4 1

Clarification of the first example:
The longest strictly increasing subsequence that can be obtained is of length 1 and there are 4 of them.
The first possible construction: writes down the first 1, the second 1 to the right: the obtained sequence is 1,1; there are two strictly increasing subsequences of length 1: **1** 1 and 1 **1**.
The second possible construction: writes down the first 1, the second 1 to the left: the obtained sequence is 1, 1; there are two strictly increasing subsequences of length 1: **1** 1 i 1 **1**.

Clarification of the second example:
The longest strictly increasing subsequence that can be obtained is of length 4.
It can be obtained only if he constructs the sequence 1 2 3 4. In that construction, it is the only strictly increasing subsequence of length 4, so the number of such is 1.

Do you know what is the difference between a hotel and a motel? That's right, the difference is in the number of flies that live there. Norman is an owner of one of the more popular American motels, but his mother insists he turns it into a hotel. That's exactly why Norman got a flyswatter (a fly-killing device) in the shape of a polygon with K edges as a Christmas present in 2016.

Wanting to meet his mother's demands, Norman found himself in front of a window with N flies. Since Norman wouldn't even harm a fly, he wants to know how many ways there are for him to hit the window with the flyswatter, without harming a single fly.

The window is a rectangle with its lower left vertex placed in the center of the coordinate system. After Norman's blow to the window, the vertices of the polygon must lie on integer coordinates, and the flyswatter must be located within the window with all its area. A fly is considered hurt if it's located on the vertex, edge or within the flyswatter.

INPUT

The first line of input contains integers X_p, Y_p and N ($1 <= X_p, Y_p <= 500, 0 <= N <= X_p*Y_p$), the coordinates of the upper right corner of the window and the number of flies on the window, respectively.

Each of the following N lines contains two integers X and Y ($0 < X < X_p, 0 < Y < Y_p$), the coordinates of a fly on the window.

The following line contains the integer K ($3 <= K <= 10000$), the number of vertices of the flyswatter.

Each of the following K lines contains two integers X_i, Y_i ($-10^9 <= X_i, Y_i <= 10^9$), the i^{th} vertex of the flyswatter. The flyswatter vertices are provided in order of joining lines, so neighbouring vertices are connected by a straight line.

OUTPUT

You must output how many ways there are for Norman to hit the window with the flyswatter, without harming a single fly.

SCORING

In test cases worth 62.5% of total points, it will hold $1 <= X_p, Y_p <= 100$

SAMPLE TESTS

input	input	input

```
4 5 2
1 3
3 4
4
0 0
2 0
2 2
0 2
```

output

```
4
```

```
5 5 3
1 4
1 3
2 2
3
4 7
6 3
7 6
```

output

```
3
```

```
6 7 2
2 5
4 5
8
1 4
3 3
4 1
5 3
7 4
5 5
4 7
3 5
```

output

```
1
```

Clarification of the third example: