International Collegiate Programming Contest 2025

Latin America Championship

March 15, 2025

Warmup Session

This problem set contains 3 problems; pages are numbered from 1 to 7.

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.kt`, *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.

Problem A – Expanding STACKS!

Tired of always waiting in lines, you invented a revolutionary restaurant concept: "STACKS! Where the last customer is served first".

The restaurant operates as follows:

- There is a single line inside the restaurant.
- When a customer enters, they immediately join the back of the line.
- Whenever a stack of glazed pancakes (the only dish at STACKS!) is ready, it's served to the person at the back of the line, who then immediately devours the pancakes and leaves the restaurant.

This business model has been incredibly successful, so much so that STACKS! is beginning to expand.

In fact, you just opened the very first STACKS!+, offering two types of pancakes: glazed and savory. The new restaurant works as follows:

- There are two lines, one for each type of pancake. Each customer joins the back of the line corresponding to the type of pancake they want.
- Whenever a stack of glazed pancakes is ready, it is served to the customer at the back of the glazed pancake line, who immediately devours it and leaves the restaurant.
- Whenever a stack of savory pancakes is ready, it is served to the customer at the back of the savory pancake line, who instantly gobbles it and leaves the restaurant.

As the boss, you want to ensure your employees follow the concept and maintain your vision. Given the order in which customers come in and out of the restaurant, you need to determine whether there is an assignment of customers to lines such that the STACKS!+ concept is followed.

You can assume that whenever a customer enters the restaurant, they immediately join the back of a line, and that they leave as soon as they are served. Also, each customer visits the restaurant exactly once.

Input

The first line contains an integer N ($1 \leq N \leq 1000$) indicating the number of customers who visited STACKS!+. Each customer is identified by a distinct integer from 1 to N.

The second line contains $2N$ signed integers $X_1, X_2, \ldots, X_{2N}$ ($1 \leq |X_i| \leq N$ for $i = 1, 2, \ldots, 2N$) indicating, in chronological order, the entrance and departure of the customers. The value $X_i = +c$ denotes the entrance of customer c into the restaurant, while $X_i = -c$ represents their departure. It is guaranteed that each customer enters and leaves the restaurant exactly once, and that they do not leave before entering.

Output

Output a single line with a string of length N if there is an assignment of customers to lines such that the STACKS!+ concept can be honored. In this case the i-th character of the string must be the uppercase letter "`G`" if customer i is assigned to the glazed pancake line, and the uppercase letter "`S`" if they are assigned to the savory line. If there are multiple solutions, output any of them.

If the STACKS!+ concept cannot be honored with the given input, output the character “*” (asterisk) instead.

Sample input 1	Sample output 1

`2`
`+2 +1 -1 -2`
	`GG`

Sample input 2	Sample output 2

`2`
`+1 +2 -1 -2`
	`GS`

Sample input 3	Sample output 3

`2`
`+1 +2 -1 -2`
	`SG`

Sample input 4	Sample output 4

`3`
`+1 +2 +3 -1 -2 -3`
	`*`

Problem B – KMOP

You probably know the KMP algorithm. You may also know that “KMP” is an acronym that stands for “Knuth Morris Pratt”, who jointly published the algorithm in 1977. How do you pronounce “KMP”? Of course, you can just say “Knuth Morris Pratt”, but what about pronouncing the acronym itself? Since “KMP” is not a pronounceable word, you are forced to say the letters one by one. In this problem we are interested in pronounceable acronyms.

We need a few definitions to formalize the requirement. A phrase is a list of words and a word is a sequence of letters. Each letter is either a vowel or a consonant. Deciding whether a letter is a vowel or a consonant depends on the language and other elements. For simplicity, we say that the six letters “A”, “E”, “I”, “O”, “U” and “Y” are vowels, while all the rest are consonants. Although it is debatable whether a given word is pronounceable, we say that a word is pronounceable when it does not contain more than two contiguous consonants. For instance, “LEMPEL” is a pronounceable word, while “DIJKSTRA” is not.

Given a phrase composed of N words, an acronym for the phrase is the concatenation of N prefixes, one prefix for each word, in the order they appear in the phrase. Each prefix must have at least one and at most three letters. Your task is to determine the minimum length a pronounceable acronym can have.

As an example with $N = 3$ consider the phrase “KNUTH MORRIS PRATT”. There are 27 possible acronyms for this phrase, such as “KMP”, “KMPR”, “KMPRA”, “KMOP”, “KMOPR” and “KNUMORPRA”, among others. Some of these acronyms are pronounceable (“KMOP” and “KMOPR”), while some others not (“KMP”, “KMPR”, “KMPRA” and “KNUMORPRA”). Since the only three-letter acronym “KMP” is not pronounceable, it follows that “KMOP” is a minimum-length pronounceable acronym for the phrase.

Input

The first line contains a positive integer N indicating the number of words in the phrase.

Each of the next N lines contains a non-empty string made of uppercase letters representing a word in the phrase. Words are given in the order they appear in the phrase. The sum of the lengths of all the strings is at most 10^6.

Output

Output a single line with an integer indicating the minimum length a pronounceable acronym can have, or the character “*” (asterisk) if no pronounceable acronym exists for the phrase.

Sample input 1	Sample output 1
3 KNUTH MORRIS PRATT	4

Sample input 2	Sample output 2
3 KNUTH M PRATT	5

Sample input 3	Sample output 3

```
3
K
M
P
```
	`*`

Sample input 4	Sample output 4

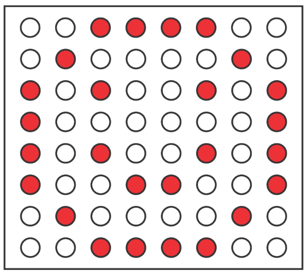
```
2
K
M
```
	`2`

Sample input 5	Sample output 5

```
4
YOU
SHOULD
BE
DANCING
```
	`5`

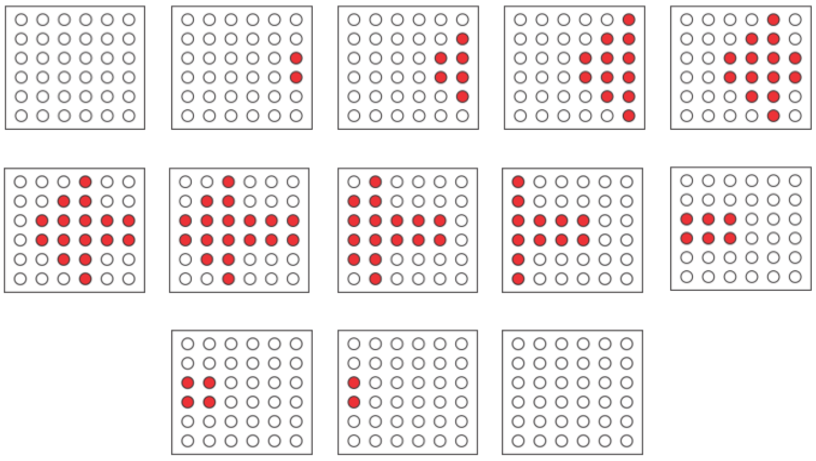
Problem C – LED Matrix

A LED matrix is a two-dimensional array of LEDs that is used to display information. This is achieved by turning on the LEDs that form a desired pattern. The figure below represents a LED matrix displaying a smiling-face pattern. LEDs that are turned off are shown in white, while LEDs that are turned on appear colored.



Some LED matrices scroll the pattern from right to left across the matrix, turning on just the appropriate LEDs at each step. Thus, any pattern with the same height than the matrix can be displayed, even patterns that are wider than the matrix. The pattern scrolling works as follows: Initially, all the LEDs in the matrix are turned off. The next step, the last column of the matrix displays the first column of the pattern. At each new step the pattern is moved one column to the left across the matrix, until the first column of the matrix displays the last column of the pattern. Finally, all the LEDs in the matrix are turned off again. If a LED matrix is equipped with pattern scrolling, the scrolling occurs even if the pattern if not wider than the matrix.

The picture below shows all the step required to display a pattern of an arrow that is pointing to the left.



Astrid has just received an old LED matrix with pattern scrolling, and she thinks that some LEDs might be broken. Since broken LEDs cannot be turned on, she is worried that some patterns will not display properly. Given the description of the state of each LED, and the pattern to display, you must tell whether the appropriate LEDs can be turned on at every step of the pattern scrolling.

Input

The first line contains three integers R, C and K ($1 \leq R, C, K \leq 1000$), indicating respectively the number of rows of both the LED matrix and the pattern, the number of columns of the matrix, and the number of columns of the pattern.

The next R lines describe the matrix and the pattern from top to bottom. Each of these lines contains a string M of length C and a string P of length K, describing respectively a row of the matrix and a row of the pattern. Each character of both M and P is either "`*`" (asterisk) or "`-`" (hyphen). For M, the character "`*`" indicates a good LED while the character "`-`" represents a broken LED. For P, the character "`*`" indicates a LED the must be turned on while the character "`-`" represents a LED that must be turned off.

Output

Output a single line with the uppercase letter "`Y`" if the appropriate LEDs can be turned on at every step of the pattern scrolling, and the uppercase letter "`N`" otherwise.

Sample input 1

```
6 6 6
****** --*---
****** -**---
****** ******
****** ******
****** -**---
*****- --*---
```

Sample output 1

```
N
```

Sample input 2

```
2 4 6
**** ------
***- *-----
```

Sample output 2

```
N
```

Sample input 3

```
2 6 4
****** ****
*-**-* ----
```

Sample output 3

```
Y
```

Sample input 4

```
1 1 1
* *
```

Sample output 4

```
Y
```

Sample input 5

```
1 1 1
* -
```

Sample output 5

```
Y
```

Sample input 6	**Sample output 6**

`1 1 1`
`- *`
	`N`

Sample input 7	**Sample output 7**

`1 1 1`
`- -`
	`Y`