icpc global sponsor
programming tools

icpc diamond
multi-regional sponsor

International Collegiate Programming Contest

2024

Latin America Championship

March 16, 2024

Warmup Session

This problem set contains 3 problems; pages are numbered from 1 to 6.

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 – Inversions

Author: Gabriel Poesia, Brasil

Every year, mathematicians and computer scientists from around the globe gather for the prestigious Inversion Counting Puzzle Contest (ICPC). For the next ICPC, the organizers had prepared the following challenge: given a string S consisting of lowercase letters, count the number of *inversions* in it. An inversion is a pair of indices $i < j$ such that S_i (the letter at position i) comes after S_j in the alphabet.

However, just last month, a group of outstanding researchers devised a sophisticated algorithm that can count the inversions in a string extremely fast. While this was great news for the advancement of science, it has been a nightmare for the ICPC staff, since their planned challenge is now obsolete.

This issue escalated to the head problem setter, who then presented a clever idea. Instead of simply receiving a string S, they should ask participants to repeat this string N times before counting the inversions. If the judges then set N to be large enough, at some point the algorithm proposed by the researchers will start to be too slow. Happy with this idea, the ICPC staff went ahead with organizing the next contest.

Unfortunately, now the judges don't know the answers to the input files anymore, and therefore can't judge submissions! The ICPC has just kicked off, and participants are about to start submitting their solutions. Please help the judges by computing the answers, so that the ICPC can run smoothly.

Input

The first line contains a string S ($1 \leq |S| \leq 10^5$), which is made of lowercase letters.

The second line contains an integer N ($1 \leq N \leq 10^{12}$) indicating how many times the string S is to be repeated.

Output

Output a single line with an integer indicating the number of inversions in the string S^N (S repeated N times). Because this number can be very large, output the remainder of dividing it by $10^9 + 7$.

Sample input 1	**Sample output 1**
`ba` `1`	`1`

Sample input 2	**Sample output 2**
`ab` `3`	`3`

Sample input 3	**Sample output 3**
`zkba` `1`	`6`

Sample input 4	**Sample output 4**
`cab` `7`	`77`

Problem B – Blackboard Game

Author: Arthur Nascimento, Brasil

Carlinhos and Equalizer are playing a game. The game begins with $3N$ elements, which are integer numbers, written on a blackboard. Then, for N rounds, the following two steps are repeated.

1. Carlinhos, the first player, selects an unchosen element and marks it with a red circle.
2. Equalizer, the second player, picks two unchosen elements, marks one of them with a blue square, and erases the other from the blackboard.

At the end of these rounds, the blackboard contains N red-marked elements and N blue-marked elements, with no moves left. The game concludes with a clear winner: if the sum of the red-marked elements differs from the sum of the blue-marked elements, Carlinhos emerges victorious; otherwise, Equalizer takes the win.

The figure below depicts the only possible outcome for the first sample. In this case Equalizer wins for sure, no matter how they play both sums will be equal to 25.

Carlinhos, feeling the game is imbalanced, seeks to determine whether he can secure a victory when both players play optimally. Can you help him with this task?

Input

The first line contains an integer N ($1 \leq N \leq 1000$).

The second line contains $3N$ integers $B_1, B_2, \ldots, B_{3N}$ ($-10^5 \leq B_i \leq 10^5$ for $i = 1, 2, \ldots, 3N$), representing the numbers initially written on the blackboard.

Output

Output a single line with the uppercase letter "`Y`" if Carlinhos can win the game and the uppercase letter "`N`" otherwise, assuming both players play optimally.

Sample input 1	**Sample output 1**
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	N

Sample input 2	**Sample output 2**
2 1 2 4 8 16 32	Y

Explanation of sample 2:
Carlinhos wins no matter how he plays, since all subsets have distinct sums.

Sample input 3	**Sample output 3**
1 2 3 3	Y

Explanation of sample 3:
Carlinhos can win by picking the number 2. Notice that he would have lost if he picked a 3.

Problem C – Deciphering WordWhiz

Author: Paulo Cezar Pereira Costa, Brasil

WordWhiz is a popular word puzzle game that challenges players to guess a secret word within a limited number of attempts. The game uses a dictionary containing N words. Each word in this dictionary consists of five distinct lowercase letters.

The game begins with the player being presented with an empty grid, consisting of a number of rows. Each row allows a single guess. The player's task is to fill rows with words contained in the dictionary until the secret word is found, or the player has used all available rows.

After the player submits a guess, the game provides feedback by coloring the cells where the guess was written. The feedback consists of three colors:

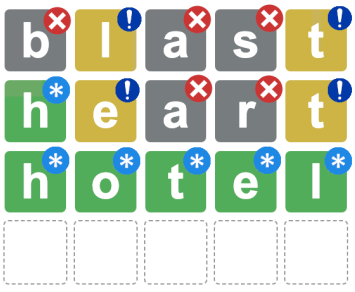
- Gray ("`X`"): The letter in the cell is not part of the secret word.
- Yellow ("`!`"): The letter in the cell is part of the secret word but is in the wrong position.
- Green ("`*`"): The letter in the cell is part of the secret word and is in the correct position.

To illustrate, let's consider the scenario where the secret word is "`hotel`", and the player submits "`blast`" as their guess. In this case, the first, third, and fourth cells would turn gray because "`b`", "`a`", and "`s`" are not present in the secret word "`hotel`". The second and fifth cells, however, would turn yellow. This indicates that "`l`" and "`t`" are part of the secret word but appear in wrong positions: "`l`" should be in the fifth position instead of the second, while "`t`" should be in the third position instead of the fifth. This feedback would be represented by "`X!XX!`".

Now, if the player submits "`heart`" as their guess, the third and fourth cells would still turn gray, because "`a`" and "`r`" are not in "`hotel`". The second and fifth cells would again turn yellow, because once more "`t`" is in the fifth position (instead of the third), and this time "`e`" is in the second position when it should be in the fourth. However, for this guess the first cell would turn green, indicating that "`h`" is the first letter in both the guess "`heart`" and the secret word "`hotel`". This feedback would be represented by "`*!XX!`".

Finally, if the player submits "`hotel`" as their guess, all cells would turn green since this is the secret word. This feedback would be represented by "`*****`".

The feedbacks above can be seen in the following picture.



Some time ago, your company added a WordWhiz player on its website and now wants to enhance the game by adding functionality to display previous game sessions. However, only the feedback for each guess was stored, not the submitted words. This means that it might not be possible to accurately recover the guesses submitted in each session, and before investing any further effort, the company wants to analyze the recorded game sessions.

Given a dictionary of five-letter words, the secret word (included in the dictionary) and the feedback for a game session, your task is to determine how many words in the dictionary could have been submitted as each guess.

Input

The first line contains an integer N ($1 \leq N \leq 1000$) indicating the number of words in the dictionary.

Each of the next N lines contains a string representing a word in the dictionary. All strings are different and each of them consists of five different lowercase letters. The first string is the secret word for the game session.

The next line contains an integer G ($1 \leq G \leq 10$) indicating the number of guesses during the game session.

Each of the next G lines contains a five-character string representing the feedback for a guess. The feedback string contains only the characters "`X`", "`!`" and "`*`", indicating respectively gray, yellow and green colors.

It is guaranteed that the input describes a valid game session.

Output

Output G lines, such that the i-th contains an integer indicating how many words in the dictionary could have been submitted on the i-th guess.

Sample input 1	**Sample output 1**
6 hotel weary heart blast pilot vague 3 X!XX! *!XX! *****	1 1 1

Explanation of sample 1:
The only possibility is that the player submitted guesses as described in the statement.

Sample input 2	**Sample output 2**
3 scale table maple 5 X!X** X!X** X!X** X!X** X!X**	2 2 2 2 2

Explanation of sample 2:
The feedback when either "`table`" or "`maple`" is submitted as a guess is "`X!X**`" (because the secret word is "`scale`"). This means that for this game session, the player could have submitted either of these words for each attempt.

Sample input 3	Sample output 3
4 scale table maple smile 4 X!X** *XX** X!X** *****	2 1 2 1

Sample input 4	Sample output 4
5 latin mrica think solve debug 1 *****	1