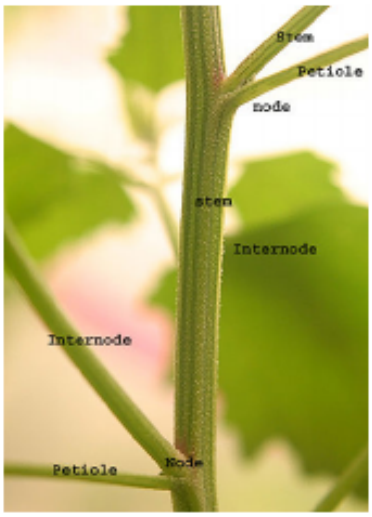
4198 Stems Sell



The designers at Spark Plug Searching, Ltd., are searching for additional ways to reduce the size of the dictionary used for their new spell checker. Their latest idea is that maybe they can get by storing only the basic "stem" form of a word without including the plural forms, past tenses, and other grammatical variants of the stem word.

When a word is extracted from a document and we want to check it for correct spelling, we can apply a series of rules to rewrite the word to its likely stem form. Then we check the dictionary to see if that rewritten word matches any entry in the dictionary.

Natural languages are pretty messy and it may take quite a bit of experimenting to get an appropriate set. So the team has proposed a simple language for describing rewrite rules. A rule has the form

pattern `=>` *replacement*

A *pattern* is a sequence of one or more characters which are interpreted as follows:

- Lower-case alphabetic characters in the pattern match a single occurrence of that same character in either upper or lower case. For example, a pattern "ab" can match and "ab", "aB", "Ab" or "AB" in a word.
- An asterisk (*) matches one or more alphabetic characters. There is at most one asterisk in a pattern, and if one is present it will be the first character in the pattern.
- An upper-case 'V' matches any lower-case vowel ('a', 'e', 'i', 'o', or 'u').
- An upper-case 'C' matches any lower-case consonant (any alphabetic character other than 'a', 'e', 'i', 'o', or 'u').
- A numeric digit 1-9 matches the same string as was matched by the character in that position of the pattern, counting the first character of the pattern as character #1. A digit k can occur in the pattern only after position k.

For example, the pattern '`*C2ies`' would match any word of at least 6 characters ending in two copies of the same consonant followed by '`ies`'.

The *replacement* part of a rule is built from a limited set of the same characters. Specifically, the replacement can contain only lower-case alphabetic characters (which are kept "as is") and numeric digits, which are replaced by the same string they would have matched in the pattern. For example, the rule

```
*C2ies => 122y
```

applied to the word "berries" would match the "*" against "be", the "C" and "2" against "r", and the "ies" against "ies". In the replacement, the "1" refers to the "*" which matched "be", the "2"s each match "r", and the "y" is left as is, so the word "berries" is rewritten to "berry".

Write a program to read a set of rewrite rules and to apply them to words in a paragraph of text. For each word (maximal consecutive string of alphabetic characters) in the paragraph, apply the rules, one at a time, in the order they are given, until a matching pattern is found or until all rules are

exhausted. If a matching pattern is found, apply that rule to rewrite the word. If not, leave the word unchanged.

Input

Input consists of multiple data sets. Each data set begins with a set of one or more rules, one per line in the format described above, followed by an empty line. This is followed by one or more lines of text, followed by an empty line or by a line containing only the left-justified string '***', which indicates end of the input

Output

For each data set, print the lines of text with each word rewritten according to the given rule set, and all non-word characters left unchanged.

At the end of the rewritten text for each data set, print a single line containing the left-justified string '***'.

Sample Input

```
*ed => 1

The quick brown foxes jumped over
the lazy yellow dogs.

*ed => 1
*ies => 1y
*s => 1

The quick brown foxes jumped over
the lazy yellow dogs.
***
```

Sample Output

```
The quick brown foxes jump over
the lazy yellow dogs.
***
The quick brown foxe jump over
  the lazy yellow dog.
***
```