A Kleene Implementation

Thor, the Norse god of thunder, was shopping for groceries when he noticed a sale on Kleenex brand tissues. This got him thinking about Kleene's recursion theorem and its application to quines in functional programming languages. As this gave him a headache, he instead turned his attention to how one might recognise regular expressions with Kleene stars on a Turing machine. Unfortunately, this just made his headache worse. So he took out a slip of paper, jotted down a brainf**k program to handle regular expressions containing Kleene plusses, paid for his groceries, and congratulated himself on a job well done.

Note: You can use any programming language you want, as long as it is brainf**k.

Input

The first line contains an integer **T** ($1 \le$ **T** ≤ 1000). Then follow **T** test cases.

For each test case: The first line contains a regular expression **P** ($1 \le$ |**P**| ≤ 30). The next line contains an integer **Q** ($1 \le$ **Q** ≤ 10). Then follow **Q** lines, each containing a string **S** ($1 \le$ |**S**| ≤ 100). Finally, there is an empty line at the end of each test case.

Each line, including the last, is terminated by a single newline (linefeed) character, which has ASCII value 10.

All regular expressions are guaranteed to be valid; in particular, **P** may not start with a plus, and it may not contain two consecutive plusses. **P** is a string over the alphabet {a,b,c,d,+}, and **S** is a string over the alphabet {a,b,c,d}.

Output

T lines each containing a string of length **Q**. The **i**th character of the string indicates whether **S** is in the regular language defined by **P**: 'Y' for a match, and '.' otherwise. Note that we are concerned whether **P** matches **S**, as opposed to a substring of **S**. In other words, we could insert '^' at the beginning of **P** and '$' at the end, and then test for a match using e.g. m// in Perl. See the example for further clarification.

Example

Input:

```
3
a
2
a
aa

a+
2
a
aa

a+bc
```

6
abbacadabba
aaaabc
abc
bc
abcd
babc

Output:

Y.
YY
.YY...

Additional Info

There are two randomly generated data sets, one with **T**=1000 and the other with **T**=500. The average value of **Q** is about 6, the probability of a match is about 0.25, the average length of **P** is about 14, and the average length of **S** is about 27.

My solution at the time of publication has 803 bytes (not golfed) and runs in 0.20s with 2.6M memory footprint.