Out of context
Problem ID: outofcontext

It's that time of year: election season. Political speeches abound, and your friend the armchair pundit likes to find quotes of politicians and use them out of context. You want to help your friend by developing a method to search through text for specified patterns.

One of the more powerful ways to express a text search pattern is using a context-free grammar (CFG). A CFG is used to generate strings, and is defined as a 4-tuple (V, Σ, R, S) where V is a set of variables, Σ is a set of terminal symbols, $S \in V$ is the starting variable, and R is a set of rules. Each rule in R is of the form

$$V \to (V \cup \Sigma)^*$$

which indicates that the head of the rule (the variable to the left of the arrow) can be replaced whenever it appears by the rule's production, a sequence of variables and terminal symbols on the right side of the arrow. It is possible for the right side of a rule to be empty. This indicates that the variable on the left can be replaced by the empty string.

A grammar generates a string of terminals by the process of *derivation*. A derivation begins with a sequence that is just the start variable. Then, until all variables have been removed, we repeatedly replace any variable in the current sequence by any one of that variable's rules.

As an example, here are rules for a grammar with start variable A (left), and an example derivation (right).

$$\begin{aligned} A &\to CFG \\ C &\to CC \\ C &\to \texttt{context} \\ F &\to \texttt{free} \\ F &\to FF \\ G &\to \texttt{grammar} \end{aligned} \qquad \begin{aligned} A &\Rightarrow CFG \\ &\Rightarrow CCFG \\ &\Rightarrow C\texttt{context}FG \\ &\Rightarrow C\texttt{context}FFG \\ &\Rightarrow C\texttt{context}FF\texttt{grammar} \\ &\Rightarrow C\texttt{contextfree}F\texttt{grammar} \\ &\Rightarrow \texttt{contextcontextfree}F\texttt{grammar} \\ &\Rightarrow \texttt{contextcontextfreefreegrammar} \end{aligned}$$

Write a program that can search through text for substrings that could have been generated by a given CFG.

Input

For this problem, V is the set of English uppercase alphabet letters and Σ is the set of English lowercase alphabet letters. Input begins with a description of R, the rules of the CFG. The first line contains an integer $1 \leq n \leq 30$, indicating the number of rules to follow. The next n lines each describe one rule, in the format

$$\texttt{[A-Z] -> [a-zA-Z]}^*$$

That is, an uppercase letter, a single space, an arrow, a single space, and a string of zero or more uppercase or lowercase letters. Rule productions are at most 10 characters long. The start variable S is the head of the first rule given.

Following the rules are at most 100 lines of text to search, each of length at most 50 characters. Each line of input consists of only English lowercase letters and spaces. Input ends at end of file.

Output

For each line to search, print a line giving the longest non-empty substring that the grammar can generate. If there is a tie for the longest, give the substring that appears earliest on the line. If there is no matching substring, print `NONE` in capital letters.

Sample Input 1

```
5
S -> aSa
S -> bSb
S -> a
S -> b
S ->
where are the abaaba palindromes on this line
none on this line
how about this aaaaaaabbbbbbbbbbbbbbbba
even a single a or b is a palindrome
```

Sample Output 1

```
abaaba
NONE
abbbbbbbbbbbbbbbbba
a
```

Sample Input 2

```
5
P -> AM
P -> M
A -> NNNx
M -> NNNxNNNN
N -> n
my phone number is nnnxnnnxnnnn what is yours
my number is nnnxnnnn at work
thanks and just in case you can also call
nnnxnnnxnnnn or nnnxnnnxnnnn
```

Sample Output 2

```
nnnxnnnxnnnn
nnnxnnnn
NONE
nnnxnnnxnnnn
```