



# Problem K

## Suffix-Replacement Grammars

Input file: suffix.in

As computer programmers, you have likely heard about regular expressions and context-free grammars. These are rich ways of generating sets of strings over a small alphabet (otherwise known as a formal language). There are other, more esoteric ways of generating languages, such as tree-adjoining grammars, context-sensitive grammars, and unrestricted grammars. This problem uses a new method for generating a language: a suffix-replacement grammar.

A suffix-replacement grammar consists of a starting string  $S$  and a set of suffix-replacement rules. Each rule is of the form  $X \rightarrow Y$ , where  $X$  and  $Y$  are equal-length strings of alphanumeric characters. This rule means that if the suffix (that is, the rightmost characters) of your current string is  $X$ , you can replace that suffix with  $Y$ . These rules may be applied arbitrarily many times.

For example, suppose there are 4 rules  $A \rightarrow B$ ,  $AB \rightarrow BA$ ,  $AA \rightarrow CC$ , and  $CC \rightarrow BB$ . You can then transform the string  $AA$  to  $BB$  using three rule applications:  $AA \rightarrow AB$  (using the  $A \rightarrow B$  rule), then  $AB \rightarrow BA$  (using the  $AB \rightarrow BA$  rule), and finally  $BA \rightarrow BB$  (using the  $A \rightarrow B$  rule again). But you can also do the transformation more quickly by applying only 2 rules:  $AA \rightarrow CC$  and then  $CC \rightarrow BB$ .

You must write a program that takes a suffix-replacement grammar and a string  $T$  and determines whether the grammar's starting string  $S$  can be transformed into the string  $T$ . If this is possible, the program must also find the minimal number of rule applications required to do the transformation.

### Input

The input consists of several test cases. Each case starts with a line containing two equal-length alphanumeric strings  $S$  and  $T$  (each between 1 and 20 characters long, and separated by whitespace), and an integer  $NR$  ( $0 \leq NR \leq 100$ ), which is the number of rules. Each of the next  $NR$  lines contains two equal-length alphanumeric strings  $X$  and  $Y$  (each between 1 and 20 characters long, and separated by whitespace), indicating that  $X \rightarrow Y$  is a rule of the grammar. All strings are case-sensitive. The last test case is followed by a line containing a period.

### Output

For each test case, print the case number (beginning with 1) followed by the minimum number of rule applications required to transform  $S$  to  $T$ . If the transformation is not possible, print `No solution`. Follow the format of the sample output.

#### Sample Input

```
AA BB 4
A B
AB BA
AA CC
CC BB
A B 3
A C
B C
C B
.
```

#### Output for the Sample Input

```
Case 1: 2
Case 2: No solution
```