Problem B: Language Cardinality

A (formal) language is a set of strings. One way to define a particular langauge is using ordinary set notation. Alternatively, some form of grammar may be more convenient for representing large sets. The UW grammar in which we are interested has two parts:

- An initial string
- A set of replacement rules of the form s_1 -> s_2 where s_1 and s_2 are strings

The language defined by this grammar is the set of all strings that can be generated by repeatedly replacing s_1 by s_2 within the initial string. For example, consider the grammar G consisting of the initial string

```
"AyB"
```

and the replacement rules

```
{"A"->"ab", "Ay"->"cdy", "B"->"w", "B"->"x"} .
```

G generates the language

```
L = {"AyB", "Ayw", "Ayx", "abyB", "abyw", "abyx", "cdyB", "cdyw", "cdyx"}
```

Given a UW grammar G, compute how many different strings there are in the language generated by G.

The first line of input contains the initial string. The second and subsequent lines contain the replacement rules, one per line, terminated by end-of-file. There are at most 100 replacement rules. Each input string contains between 0 and 10 upper and lower case letters, and is enclosed in quotes. There are no spaces in the input.

Output consists of a single integer, the number of distinct strings in the language generated by G. If there are more than 1000 distinct strings, print "Too many." instead.

Sample Input

```
"AyB"
"A"->"ab"
"Ay"->"cdy"
"B"->"w"
"B"->"x"
```

Output for Sample Input