6280 Disjoint Regular Expressions

Mark is developing new social network *Facepalm* for inhabitants of Phobos and Deimos. His recent task is to add information about home asteroid of the owner to each account. Of course, each account owner could enter such information, but Mark decided that it would be more convenient if some default value was suggested to the user at logon. He investigated the situation and found out that home asteroid of a user can be found by analyzing his last name.

Last name of each user of Facepalm is a non-empty word consisting of lowercase letters of the English alphabet. Users from Phobos have their last names matching regular expression P while users from Deimos have their last names matching regular expression D.

However, the problem is that some last names can match both expressions. Two expressions are called *disjoint* if there is no such non-empty string s that matches both expressions. Mark believes that expressions P and D are disjoint. However he needs your help to check it.

You are given two regular expressions P and D. Check whether they are disjoint, and if they are not, find the shortest non-empty string s that matches both of them. If there are several shortest common strings, you can find any one.

Note: Let us define regular expressions and matching strings to them formally.

- A single lowercase letter c is a regular expression, it matches only a string consisting of a single letter c.
- Alternation: if P and Q are regular expressions then $(P|Q)$ is a regular expression, a string α matches it if α matches P or α matches Q.
- Concatenation: if P and Q are regular expressions then (PQ) is a regular expression, a string α matches it if $\alpha = \beta\gamma$, β matches P and γ matches Q.
- Kleene star: if P is regular expression then $(P*)$ is a regular expression, a string α matches it if α can be represented as a concatenation of zero or more strings $\alpha_1\alpha_2\ldots\alpha_k$ where all α_i match P. An empty string always matches Kleene star.

Parentheses can be omitted, in this case Kleene star has the highest priority, then concatenation and then alternation. For example, "`abc*|de`" means "`(ab(c*))|(de)`".

Input

The input will contain several test cases, each of them as described below.

The input contains two lines. The first line contains regular expression P. The second line contains regular expression D. Each expression contains from 1 to 100 characters.

Output

For each test case, the output must follow the description below.

If Mark's guess is correct and the two expressions are indeed disjoint, print "`Correct`" at the first l ine of the output file. If they are not, print "`Wrong`" at the first line of the output file. In this case the second line must contain any shortest non-empty string that matches both expressions.

Sample Input

```
a(ab)*b
a(a|b)*ab
a(ab)*a
a(a|b)*ba
```

Sample Output

```
Correct
Wrong
aaba
```