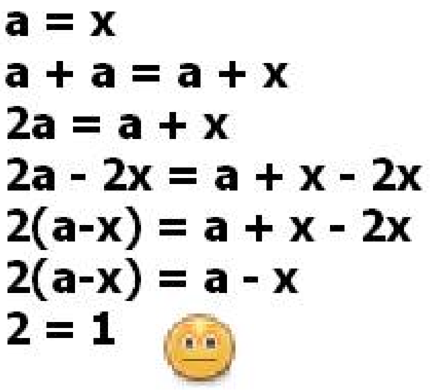
In this problem, we will consider a mathematical expression according to the following BNF grammar:

$< expression > $ `=` $< variable >$ `|` $< expression >< operator >$
$< expression >$ `|` `"("`$< expression >$`")"`
$< operator >$ `= "+" | "*"`
$< variable >$ `= "a" | "b" | "c" | ... | "y" | "z"`



When evaluating such expressions, you have to follow the conventional rules. That means you have to do things in the brackets first and multiplications have to be done before addition.

Example: 2*(3+4*2) = 22

Given an expression and some inequalities, you have to assign each variable with a positive integer so that the value of the expression is minimized.

Consider an example:
Expression = a+b*c and Inequalities = a¿b, c¿b
Assignment of: $a = 2$, $b = 1$ and $c = 2$ will give us the minimum value. `=> 2 + 1*2 = 4`

Input

The first line of input is an integer T ($T < 100$) that gives us the number of test cases. Each case starts with a line that gives you the expression. The length of the expression is at most 300. The next line contains an integer I ($I < 400$) that gives you the number of inequalities. Each of the next I lines will give you an inequality of the format $x > y$ where x and y are lowercase alphabets that are present in the given expression and x is not equal to y. All the inequalities will be distinct.

Output

For each case, output the case number first. Then output the minimum value of the expression that can be obtained by assigning positive integers to each variable that abides by the given inequalities. You can assume the output will fit in 32 bit signed integer. If the inequalities are inconsistent, then output '`-1`' instead.

Sample Input

```
3
a+b*c
2
a>b
c>b
z*(x+y)
3
z>x
x>y
z>y
a+b+c+a
0
```

Sample Output

```
Case 1: 4
Case 2: 9
Case 3: 4
```