Let us define GNU, the recursive acronym for GNU's Not Unix with the following recursive rules:

1. G –> GNU's
2. N –> Not
3. U –> Unix

In each step we apply all the rules simultaneously. If a character in a string does not have a rule associated with it (there will be at most one rule per character), it remains in the string.

For example if we start with GNU, we get:

Step	String
0	GNU
1	GNU'sNotUnix
2	GNU'sNotUnix'sNototUnixnix
3	GNU'sNotUnix'sNototUnixnix'sNototototUnixnixnix
4	GNU'sNotUnix'sNototUnixnix'sNototototUnixnixnix'sNototototUnixnixnixnix
...	...

As you can see, the strings are growing larger in every steps. And in certain cases the growth can be quite fast. Fear not, for we're not interested in the entire string. We just want to know the frequency of a particular character after a finite number of steps.

Input

The first line of the input starts with an integer, T ($1 \leq T \leq 10$), the number of test cases. Then T test cases will follow. The first line of each test case will give you R, ($1 \leq R \leq 10$) the number of rules. In each of the next R lines there will be one rule. The rules are written in the following format: 'x->S' (without the quotes). Here x is a single character and S is a sequence of characters. You can assume that the ASCII value of the characters lie in the range 33 to 126, that S will contain no more than 100 characters. You can also assume that the symbols '-' and '>' won't appear in S. The rules can be immediately recursive, recursive in a cycle or not recursive at all. After the rules, you'll find an integer Q ($1 \leq Q \leq 10$), the number of queries to follow. Each of the Q queries will be presented in the format "*initial_string* x n" in a single line, where *initial_string* is the string that you have in step 0 (with length between 1 and 100), x is the character whose frequency you should count, and n ($1 \leq n \leq 10000$) is the number of times the rules should be applied. All characters in the initial string will have ASCII values in the range 33 to 126.

Output

For each query, print the number of occurances of the particular character in the result string after applying the rules n times, starting with the initial string. The output will always fit in a 64-bit unsigned integer.

Sample Input

```
2
3
G->GNU's
N->Not
U->Unix
2
GNU t 3
GNU N 3
1
A->BAcX
1
ABCcXA c 10000
```

Sample Output

```
6
4
20001
```