**Problem Statement:**

Given n regular expressions r1, r2, . . . , rn and a string w, the LEX tool finds the longest prefix w1 (say w1 as a valid token) of the input w that can be generated by a regular expression (RE) in {r1, r2, . . . , rn}. If the longest prefix w1 can be generated by more than one regular expression, consider that it is generated by the lower indexed regular expression among the REs that can generate w1 and output the index of the regular expression preceded by a $, i.e., if w1 can be generated by r3, r5, and r8, then consider that w1 is generated by r3 and outputs $3. Now, repeating the process with the remaining part of the input w. If a character is not part of any valid token, it echoes and outputs the character preceded by a @. Therefore, each character of w is either part of a valid token or echoed.

Here is a file containing C / C++ code without using <regex> library that takes n regular expressions r1, r2, . . . , rn and a string w and outputs the sequences of valid tokens, as per LEX tool, and echo if a character is not part of any token. Here basically have generated NFA’s for +,\*,concatenation and union and then match it using dfs.

Here the tokens are actual lexemes.

Further note the following:

1. Assume that Σ = {a, b} is the alphabet set i.e., any regular expression r (as given as input) will generate a language L(r) ⊆ {a, b} ∗ .

2. Operations on the regular expressions: concatenation, union, closures (both ∗ and +).

Input format: A text file, input.txt contains n + 2 lines; the first line contain the value of n, the lines from 2, 3, . . . , n + 1 contains n regular expressions r1, r2, . . . , rn (one per line) and the (n + 2)-th line contains the input string w.

Please note that each sub-regular expression is parenthesized.

Output format: output.txt contains the output for the given instance.

Example : content of input.txt:

2

(((a)∗)(b))

((b)(a))

abbaababaaabbaa

content of output.txt: $1$2$1$1$1$2@a#