CS F363 Compiler Construction
Assignment-1 (Question-1)

Due date: 17 Feb 2024 11:59 PM **Marks : 14** (7%)

In this assignment, you must write a C/C++ program that works like the LEX tool. The formal description of the problem statement is given below. Given k regular expressions $r_1, r_2, \ldots, r_k$ (in the same order) and a string w, the LEX tool breaks the input w into a sequence of longest lexemes that can be generated by a regular expression from $r_1, \ldots, r_k$. It will be echoed if any regular expression cannot generate a character in w.

Your task is to write a C / C++ code that takes k regular expressions $r_1, r_2, \ldots, r_k$, and a string w as an input and output the sequences of longest lexemes along with the index of the regular expression that can generate the lexeme. Note that if more than one regular expression can generate the same lexeme, then output the index of the lowest indexed regular expression that can generate the lexeme, as per the LEX tool, and echo if a character cannot generated by any regular expression and, in this case, output the index as 0.

Further, note the following:

1. Assume that $\Sigma = \{a, b\}$ is the alphabet set i.e., any regular expression r (as given as input) will generate a language $L(r) \subseteq \{a, b\}^*$.
2. Operations on the regular expressions: concatenation, union, closures (both $*$ and $+$), and at most one occurrence (?).

Input format: The input is read from a text file, named *input.txt* and the file contains the input string w in the first line and contains a sequence of regular expressions $r_1, r_2, \ldots, r_k$, one per line. Note that k is not part of the input.
Please note that each sub-regular expression is parenthesized, as discussed in Labsheet 2.

Output format:
Generate a file, output.txt (do not use other names), that contains one string that represents the output and string is of the form $< \mathsf{lexeme_1}, \mathsf{t_1} >< \mathsf{lexeme_2}, \mathsf{t_2} > \cdots < \mathsf{lexeme_n}, \mathsf{t_n} >$ (do not use any extra spaces) where $\mathsf{lexeme_1}$ is the longest prefix of the input w that can be generated by regular expression r_{t_1}, and $\mathsf{lexeme_2}$ is the longest prefix of the remaining input w that can be generated by regular expression r_{t_2}, and so on. Note the following:

1. If $\mathsf{lexeme_i}$ can be generated by more than one regular expression from $\{r_1, r_2, \ldots, r_k\}$, then t_i is the lowest index among the regular expressions that can generate $\mathsf{lexeme_i}$.
2. If $\mathsf{lexeme_i}$ is not generated by any regular expression (i.e., $\mathsf{lexeme_i}$ is echoed), then t_i is 0.

Example 1:
input.txt

```
abbaababaaabbaa
(((a)*)(b))
((b)(a))
```

output.txt

```
<ab,1><ba,2><ab,1><ab,1><aaab,1><ba,2><a,0>
```

Example 2: input.txt

```
baabaababbababaaaabbbababa
((((b)|((a)(b)))+)|((((b)|((a)(b)))*)(a)))
((((a)(a))+)(b))
(((b)(((a)(a))?))(b))
```

output.txt

```
<baab,3><aab,2><abbababa,1><a,1><aab,2><bbababa,1>
```

Submission guidelines:

1. Submit a single C / C++ file and name it with your group code. If your group code is BITS1234, your file name must be BITS1234.cpp or BITS1234.c.
2. One submission per group is sufficient.
3. Strictly follow the input and output formats.
4. **Late submission policy:** Each 1hr delay will fetch 1% penalty of the total marks obtained, and late submissions will not be accepted after 48 hours from the due date.

Important Note: You cannot use <regex> library or any similar libraries.