# Practice Problems: Lexical Analyzer

**Q1**. Design the transition diagrams for relational operator (such as >, >=, >> (right shift operator), >>= (right shift assign operator) and write a manual C code implementation for the same as discussed in the class.

**Q2**. Design regular definition, transition diagram and convert into your C code for your unsigned integers. In addition, the notations may or may not include the Qualifier (unsigned (u|U) or long (l|L) or null) as a suffix at the end of the respective notations.

**Q3**. Write the regular expression for variable identifier in C language. Construct an equivalent DFA to recognize variable identifiers. Also use the DFA to recognize xyz_1, 1xyz and _1xyz strings by manually traversing the DFA. Which of these are the valid names? Which ones do not take you to the accept state?

**Q4**. Can a variable identifier in C language have any number of underscores? Can it have only the underscores as variable identifier name?

**Q5.** Execute the following code in C language main(){ int __, ____; __=3; ____=5; printf("%d", __+____);}//include necessary files yourself. Observe the compilation and execution results.

**Q6**. How are the C and Python programming languages different in describing patterns for variable identifiers, all operators, keywords, comments, separators, etc.?

**Q7.** Construct an ε-NFA that can recognize a <= (less than or equal to), < (less than) and << (left shift). Compute ε- closure of all the states and convert the ε-NFA to NFA. Also use subset construction method to convert the NFA to DFA.

**Q8**. Write the regular expressions for all keywords used in C language. A language uses names with patterns [a-c][2-5][a-c]*[2-5]* and [a-c][6-8][2-5]*[a-c]* to represent variable identifiers of integer and real types. Write a C like program to compute the roots of a quadratic equation using these identifiers. You need not declare the type of variables. Construct a DFA for such identifiers.

**Q9**. Write a C program that has 5 lexical errors. Compile the program to observe lexical errors reported with line numbers.

**Q10**. Generate a C Lexical analyser whose identifier, keyword and relational operator specifications are same as discussed in the regular lectures. However, regular definition of keywords will come first in the precedence order as compared to the identifiers. Verify the output w.r.t to your own test cases.