LAB CYCLE

1. Design and implement a lexical analyzer for C language. Program should recognize the tokens such as identifiers, keywords, arithmetic operators and relational operators (including line numbers) in a given C program.
2. Develop an operator precedence parser for the grammar below.
   E → E + E / E - E / E * E / E / E / E ^ E / (E) / id
3. Construct a recursive descent parser for an expression according to the grammar below.
   E → TE'
   E' → +TE'/- TE'/ε
   T → FT'
   T' → *FT'/ /FT'/ ε
   F → (E) / id
4. Write program to find First and Follow of any given grammar.
5. Implement Intermediate code generation for a given set of arithmetic expressions in triple and quadruple format.
6. Implement the back end of the compiler which takes the three-address code and produces the 8086 assembly language instructions that can be assembled.
7. Write a program to perform loop unrolling.
8. Optimize intermediate code by Common Subexpression Elimination of basic blocks.
9. Write program to find ε-closure of all states of any given NFA with ε transition.
10. Write program to convert NFA to DFA.
11. Implementation of Lexical Analyzer using Lex Tool to recognize tokens such as identifiers, keywords, operators.
12. Generate YACC specification for the implementation of simple calculator that performs +, –, * and /.