Compiler Design Lab
Lab performance evaluation

Programs to push into Github

1. Implement lexical analyzer using C for recognizing the following tokens:
   - A minimum of 10 keywords of your choice
   - Identifiers with the regular expression : letter(letter | digit)*
   - Integers with the regular expression: digit+
   - Relational operators: <, >, <=, >=, ==, !=
   - Storing identifiers in symbol table.
   - Using files for input and output.
2. Implement lexical analyzer using LEX for recognizing the following tokens:
   - A minimum of 10 keywords of your choice
   - Identifiers with the regular expression : letter(letter | digit)*
   - Integers with the regular expression: digit+
   - Relational operators: <, >, <=, >=, ==, !=
   - Ignores everything between multi line comments (/* …. */)
   - Storing identifiers in symbol table
   - Using files for input and output.
3. Construct Recursive Descent Parser for the grammar G = ({S, L}, {(,), a, ,}, {S → (L) | a ; L→ L, S | S}, S) and verify the acceptability of the following strings:
   i. (a,(a,a))
   ii. (a,((a,a),(a,a)))

   You can manually eliminate Left Recursion if any in the grammar.

4. Implement Predictive Parser using C for the Expression Grammar

   E → TE'
   E'→ +TE' | ε
   T → FT'
   T'→ *FT' | ε
   F → (E) | d
5. Implementation of Shift Reduce parser using C for the following grammar and illustrate the parser's actions for a valid and an invalid string.

   S –> 0S0 | 1S1 | 2
6. Implement LALR parser using LEX and YACC for the following Grammar:

   E → E+T |T
   E'→ T*F | F
   F → (E) | d

7. Implement LALR parser using LEX and YACC for the following Grammar by specifying proper precedence for operators:

   E → E+E | E-E | E*E | E/E | -E | (E) | digit

8. Generate quadruples for given arithmetic expression using LEX and YACC.