LIST OF EXPERIMENTS

1. Write a C program to implement Recursive Descent parser for the following grammar

S → (L)/a

L → SL’

L’ → ,SL’/∈

1. Implement C program to demonstrate shift reduce parsing technique the grammar:

E → E + E/E\*E/ E/E /a/b

1. Write a C program to implement Predictive parser for the following grammar

E → TE’

E’→ +TE’/∈

T → FT’

T’→ \*FT’/∈

F → (E)/i

1. Write a C program to implement symbol table.
2. Develop a lexical analyzer to identify identifiers, constants, operators using C program.
3. Write a Lex program to implement a simple calculator.
4. Implement the back end of the compiler using C program.
5. Write a C program to identify whether a given line is a comment or not
6. Write a C program to test whether a given identifier is valid or not.
7. Write a C program to validate operators.
8. Write a LEX specification file to take input C program from a .c file and count the number of characters, number of lines & number of words.
9. Write a LEX program to count the number of Macros defined and header files included in the C program.

13. Write a LEX program to print all the constants in the given C source program file.

14. Write a LEX program to print all HTML tags in the input file.

15. Write a LEX program to count the number of comment lines in a given C program and eliminate them and write into another file.

1. Write a LEX program to count the number of comment lines in a given C program and eliminate them and write into another file.
2. Implement a C program to eliminate left recursion and left factoring from a given CFG.
3. Write a C program to find FIRST and FOLLOW for predictive parser.
4. Write a C program for constructing of LL (1) parsing.
5. Write a C program to implement operator precedence parsing.
6. Implement a simple intermediate code generator in C program, which produces three address code statements for a given input expression.
7. To implement the intermediate code generation using Lex and for control flow while loop statement.
8. Implement simple code optimization techniques using Cprogram.
9. Find the topological order of a diagraph G represented as adjacency lists
10. Implement storage allocation strategies using stack.