Compiler Design Lab: List of Programs

1. Write a Case Study on GCC and C Language for different stages of compilation, see different file types at stages, etc.
2. Write a C Program to Scan and Count the number of characters, words, and lines in a file.
3. Write a C Program Count total No. of Keywords in a File.
4. Write a C Program Count Total no of Operators in a File.
5. Write a C Program Count Total no of occurrence of each character in a File.
6. Write a C Program to To identify whether given string is keyword or not
7. Write a C Program to implement DFAs that recognize identifiers, constants, and operators of the C language.
8. Write a program in LEX to recognize Floating Point Numbers
9. Write a program in LEX to recognize different tokens: keywords, Identifiers, constants, operators and punctuation symbols
10. Write a program in LEX to count vowels and constants in a C file
11. Write a program in LEX that copies file, replacing each nonempty sequence of white spaces by a single blank.
12. Write a C program that takes another C program as input and
    a. strip the spaces, remove the comments
    b. Use regex to identify the patterns for identifier, keywords, punctuation, operators
    c. Identify the lexeme for identifier, keywords, punctuation, operators
    d. Generate Symbol Table for each new identifier to store related attributes.
    e. Generate stream of tokens for Syntax Analysis phase.
13. Design a Lexical analyzer for the above language. The lexical analyzer should ignore redundant spaces, tabs and newlines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value.
14. Write a C program to identify that grammar is ambiguous.
15. Write a C program to identify that grammar is Left Recursive.
16. Write a C program to identify that grammar is non deterministic.
17. Write a C program to apply left factoring to a non deterministic grammar to generate equivalent deterministic grammar.

18. Write a C program to perform Brute Force parsing with backtracking.
19. Write a C program to perform Recursive Descent parsing.
20. Write a C program to perform LL(1) parsing, including generateFirstList(), generateFollowList(), generateParsingTable(), parse(), etc. functions.
21. Write a C program to perform LR(0) parsing, including generateCanonicalListItems(), closure(), goTo(), generateParsingTable(), parse(), etc. functions.
22. Use YACC to Convert Binary to Decimal (including fractional numbers).
23. Use YACC to implement, evaluator for arithmetic expressions
24. Use YACC to convert: Infix expression to Postfix expression.
25. Design Predictive parser for the given language
26. Design LALR bottom up parser for the above language.
27. Convert the BNF rules into Yacc form and write code to generate abstract syntax tree.
28. Write a program to generate machine code from the abstract syntax tree generated by the parser. The following instruction set may be considered as target code.