University of Engineering & Management, Kolkata
Even Semester, 2023
Course: B. Tech Semester: 6th
Paper Name: Compiler Design Laboratory
Paper Code: PCC-CSE691

List of Experiments

Week 1
Programs on the following topic:

Develop a lexical analyzer to recognize a few patterns in C. (Ex. identifiers, constants, Comments, operators etc.)

1. Write a C program to check if a user given string is a valid identifier or not?
2. Write a C program to check if a user given C program statement is a valid Comment or not?
3. Write a C program to read a program written in a file and remove all comments. After removing all comments, rewrite the program in a separate file.
4. Write a C program to convert an infix statement into a postfix statement.
5. Write a C program to evaluate an arithmetic expression which is given as a string. Consider the input has no parentheses and contains the following operators only: +, -, *, /

Week 2
Programs on the following topic:

Implementation of Lexical Analyzer using Lex Tool

6. Write a Lex Program to count the number of vowels and consonants in a given string
7. Write a Lex Program to count the number of characters, words, spaces, end of lines in a given input file.
8. Write a Lex Program to count no of: a) +ve and –ve integers b) +ve and –ve fractions
9. Write a Lex Program to count the no of comment line in a given C program. Also eliminate them and copy that program into separate file.
10. Write a Lex Program to count the no of ‘scanf’ and ‘printf’ statements in a C program. Replace them with ‘readf’ and ‘writef’ statements respectively
11. Write a Lex Program to recognize a valid arithmetic expression and identify the identifiers and operators present. Print them separately.
12. Write a Lex Program to recognize whether a given sentence is simple or compound.
13. Write a Lex Program to implement arithmetic calculator.

Week 3
Programs on the following topic:

Generate YACC specification for a few syntactic categories.

a) **Program to recognize a valid arithmetic expression that uses operator +, –, * and /.**

b) Program to recognize a valid variable which starts with a letter followed by any number of letters or digits.
c) Implementation of Calculator using LEX and YACC

14. Write a Lex Program to recognize and count the number of identifiers in a given input file.
15. Write a YAAC Program to test the validity of a simple expression involving operators +, -, * and /
16. Write a YAAC Program to recognize nested IF control statements and display the levels of nesting.
17. Write a YAAC Program to check the syntax of a simple expression involving operators +, -, * and /
18. Write a YAAC Program to evaluate an arithmetic expression involving operating +, -, * and /
19. Write a YAAC Program to recognize a valid variable, which starts with a letter, followed by any number of letters or digits.
20. Write a YAAC Program to recognize strings ‘aaab’, ‘abbb’, ‘ab’ and ‘a’ using grammar (an b n , n>=0)
21. Write a YAAC Program to recognize the grammar (an b, n>=10)
22. Write a YACC Program to implement arithmetic calculator.

Week 4
Programs on the following topic:
Implementation of Symbol Table

23. Write a Program to implement Symbol Table.

Week 5
Programs on the following topic:
Convert the BNF rules into YACC form and write code to generate Abstract Syntax Tree

24. Write a Program to Convert the BNF rules into YACC form and write code to generate Abstract Syntax Tree

Week 6
Programs on the following topic: Implement type checking

25. Write a C program to implement type checking.

Week 7
Programs on the following topic:
Implement control flow analysis and Data flow Analysis

26. Write a C program to implement control flow analysis and Data flow Analysis.

Week 8
Programs on the following topic:
Implement any one storage allocation strategies (Heap, Stack, and Static)

27. Write a C program to implement Stack storage allocation strategies.

Week 9
Programs on the following topic:
Construction of DAG

28. Write a C program to implement DAG.

Week 10
Programs on the following topic:
Implement the back end of the compiler

29. Implement the back end of the compiler which takes the three address code and produces the 8086 assembly language instructions that can be assembled and run using a 8086 assembler. The target assembly instructions can be simple move, add, sub, jump. Also simple addressing modes are used.