LAB Assignments

Week 1
Date: 03-01-2022

1. Identify the files created while compilation and execution of a C program.
2. Create a C program for generating a Lexical Analyzer to identify the following tokens:

Keywords; Examples-for, while, if etc.

Identifier; Examples-Variable name, function name, etc.

Operators; Examples '+', '++', '-' etc.

Separators; Examples ',' ';' etc.

Week 2
Date: 10-01-2022

1. Install FLEX.
2. Create a C program using FLEX for generating a Lexical Analyzer to identify the following tokens:

   Keywords; Examples-for, while, if etc.

   Identifier; Examples-Variable name, function name, etc.

   Operators; Examples '+', '++', '-' etc.

   Literals; (you may identify, INT, FLOAT, REAL)

   Separators; Examples ',' ';' etc.

Note: Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value.

Week 3
Date: 17-01-2022

1. Write a program using FLEX to read and validate a mathematical expression and display the result. The result should follow the BODMAS Rule. If the expression is invalid then display as invalid.
2. Write a program using FLEX to count the number of:
   (a) Lines
   (b) Words
   (c) Capital Letters
   (d) Small Letters
   (e) Numbers (10,21)
   (f) Digits (0-9)
   (g) Special Character
   (h) Delimiter
   (i) Operator
   (j) Relational Operator
   (k)Total Characters

Week 4
Date: 24-01-2022

1. Consider the simplest regular expressions which are the strings of text characters with no operators at all like **monday tuesday wednesday**. These three regular expressions match any occurrences of those character strings in an input text. Design a Scanner that removes every occurrence of the word **day** in such words.

2. Design a Lexical analyzer which is successfully able to execute the following tasks:

(i) Count the total number of all digit strings in an input text
(ii) Print the running total of the number of digit strings
(iii) Print out each one as soon as it is found.

3. Consider some hypothetical pseudo code instructions issued for a robot in order to carry out the horizontal and vertical motions like LEFT, RIGHT etc. Identify tokens for 5 such instructions issued (of your choice).

Week 5 & Week 6
Date: 31-01-2022 and Date: 07-02-2022

1. Write a C program that accepts a grammar and removes left recursion from it.

2. Write a C program that accepts a grammar and removes left factoring from it.

3. Submit the synopsis of your project as per discussion and instructions in Theory class.

(Hint: You may refer the sample project uploaded under Project folder in Classroom)

Week 7

Date: 21-02-2022

1. Write a C program to find First and Follow for all non-terminals.
2. Explore Yacc Tool.

Refer:

https://softadvice.informer.com/Yacc_0.4.0.3_Installer.exe_Download.html

https://www.ibm.com/docs/en/zos/2.4.0?topic=tools-generating-parser-using-yacc

Week 8

Date: 28-02-2022

1. Download and Install Yacc Tool on your system and prepare a lab manual.

   Refer:

   https://softadvice.informer.com/Yacc_0.4.0.3_Installer.exe_Download.html

   https://98h.org/57-how-to-install-lex-and-yacc-in-ubuntu.html

   https://simran2607.medium.com/compiler-design-using-flex-and-bison-in-windows-a9642ebd0a43

   http://csbeans.blogspot.com/2015/01/how-to-compile-lex-and-yacc-in-windows.html

2. Write a Yacc program to check if entered statement is a valid arithmetic expression

Week 9

Date: 21-03-2022

1. Design a grammar for a declarative statement for C program. Further, write a Yacc program to check if the entered statement is a valid declarative statement according to the grammar generated.
2. Design a grammar for a relational expression of C language. Further, write a Yacc program to check if the entered statement is a valid relational expression according to the grammar generated.
3. Design a grammar for a logical expression of C language. Further, write a Yacc program to check if the entered statement is a valid logical expression according to the grammar generated.

Week 10 & Week 11

Date: 28-03-22 & 04-04-22

1. Consider the example of simple desk calculator that performs simple operations on integer expressions with the grammar:

exp -> exp addop term | term addop -> + | -

term -> term mulop factor | factor mulop -> *

factor -> (exp) | number

number -> number digit | digit

digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

You are required to write YACC specifications for this grammar so that the parser evaluates any arithmetic expressions and the output shows each grammar rule as it is applied in the parsing process. Show your parsing sequence for the input string: (2+(3*4))

2. The following grammar describes a boolean expression (exp) consisting of operators "&", "|", "!", "==", "! =", brackets "(" and ")", and identifiers.

exp : exp_2 | exp '&' exp_2 ;

exp_2 : exp_3 | exp_3 '|' exp_2 ;

exp_3 : exp_4 | exp_4 '==' exp_4 | exp_4 '!=' exp_4 ; exp_4 : exp_5 | '!' exp_5 ;

exp_5 : identifier | '(' exp ')' ;

Write Yacc code to recognize a series of expressions, each on a new line. Each expression should unambiguously demonstrate precedence & associativity of the operators, by showing the orders in which operators would be evaluated. Also, give a parse sequence for each of the expressions.

3. Using YACC, write semantic actions that translate arithmetic expressions (generated from the given grammar) from infix into postfix notation

E -> E + T E -> T

T-> T * F T -> F

F -> (E) F -> num

4. For the grammar below, write a YACC program to compute the decimal value of an input string in binary. For example the translation of the string 101.101 should be the decimal number 5.625.

S→L.L|L L→LB|B B→0|1

5. Write a YACC program that accepts valid strings as per the grammar given below and produces as output a linear representation of the same list

Grammar:

S (L)| char

L L,S|S

and char is any character between a-z

e.g. ((a,c),d,(h)) Produces acdh but (a,c),d,(h)) is an invalid input

Week 12 and Week 13

Date: 18-04-22 & 25-04-22

1. Write a yacc program to check if the parenthesis are balanced and count the number of matching parenthesis

   P->(P) | a

2. Write a YACC program to recognize expressions involving binary numbers as operands and print the decimal value of the expression

   > 1001

   Decimal value: 9

   >101 + 100

   Decimal value: 9

   > 201 + 101

   Invalid input.

3. Append semantic actions to the grammar of simple desk calculator that performs simple operations on integer expressions with the grammar:

   exp -> exp addop term | term

   addop -> + | -

   term -> term mulop factor | factor

   mulop -> *

   factor -> (exp) | number

   number -> number digit | digit

   digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

   You are required to write YACC specifications for this grammar so that the parser evaluates any arithmetic expressions and the output shows each semantic action as it is applied in the parsing process. Show your parsing sequence for the input string:

   (2+(3*4))

4. Generate a parser for sql select statement (select statement may contain group by, having clause)and do the type checking wherever required. Count the number of attributes selected and number of attributes used in conditions.

For eg.: select name,address from emp where age>20 group by name having age<40 order by name desc;

5. Write a YACC program to evaluate the following expressions. Specify the grammar clearly.

Functions Meaning

(+ x1 x2 x3 ….. xn) Calculate Sum of x1, x2 upto xn

(* x1 x2 x3 ….. xn) Calculate Product of x1, x2 upto xn

(max x1 x2 x3 ….. xn) Calculate Maximum of x1, x2 upto xn

(min x1 x2 x3 ….. xn) Calculate Minimum of x1, x2 upto xn

Sample Input:

(+ 6 12 18)

Sample Output:

36