III B. Tech (CSE) – II Semester

CS 354 - LANGUAGE PROCESSORS LAB ASSIGNMENTS

CYCLE 1:

NOTE: Assignments 1 to 6 are to be completed before the mid-semester examinations. The assessment will be completed by the week prior to the mid semester examinations.
Assignment 7 can be completed before end of February 2019. Assessment for this assignment will be done in the first week of March 2019.

1. Implement CYK algorithm for deciding membership of a string in CFG in CNF. Assume that terminals and non-terminals are represented by a single alphabet and each grammar rule is given as a string where the first symbol is left side and the remaining portion is the corresponding right side of the production. Modify it to produce the number of non-identical derivation sequences for a given string in the grammar. Also generate the distinct derivation sequences, if possible.

2. Write flex specifications to convert an infix expression to postfix notation by constructing the corresponding expression tree. Use the binary operators +, -,*, /, @(exponentiation) and um (Unary Minus). Assume usual precedence and associativity rules for the operators. Note that some of the operator symbols are meta-characters in flex.

3. Consider a CFG with each terminal and non-terminal represented as single alphabet and each production is represented as a string with left most symbol as left side variable of the production. Write a program to eliminate useless symbols, € - productions and unit productions. From the resultant grammar remove immediate left recursion. Display the input and output grammars in a readable notation.

4. Write a program to encrypt the text using lex. Copy a text file with some predefined words as special and encrypt the text as follows:
   - Special words have to be rotated right by two positions (cyclic) and written right to left.
   - Other English words should be encrypted by replacing each character by ASCII + k-cyclic and words need to be written in reverse.
   - Each number should be replaced by a new number which is generated by swapping left half and right half of the number. If the number has odd number of digits, the middle digit should be at the same place
   - All other characters have to be retained as it is except the sequence of whitespaces which need to be replaced by single space.

5. Implement a desk calculator with the operators defined in assignment 2 along with named identifiers, assignment statements, if statement, logical expressions (<, >, ==, != for less than, greater than, equal to and not equal to as used in C language). Write a function “let” for initializing a variable and “display” to display the value of a variable or expression.

6. Write a program to generate a symbol table using words in a given English text. Also include rules to recognise words like “can’t” as a single word. Use a normal table for storing words, their frequency and unique line numbers in which the word appears.

7. Write flex and bison specifications to generate intermediate code in the form of three address code statements for a subset of C language with the following functionalities:

- Entire program consists of only main() function. There are no other functions.
- All identifiers are of integer (signed and unsigned) or float type only.
-
- All floats are represented as num.num
- In case of mixed mode, the integers are to be explicitly converted to floats.
- Variables, where ever necessary are initialized by assignment only. No values are explicitly read.
- Boolean values are considered like in C language
- Binary operators: +, -, *, /, % and @(exponentiation)
- Ternary Operator: ?:
- Assignment operators : =, +=, -=, *=, /=
- Relational operators: <, >, <=, >=, !=
- Logical operators: &&, ||, !
- Bitwise operators: |, &, !
- Conditional statements: if(E)S; and if(E) S1; else S2;
- Iterative Constructs: while(E)S; and for(e1;e2;e3)S;

The following stages of completion of this assignment are to be explicitly demonstrated:

- Grammar rules for the language and document the grammar properly.
- Flex and Bison specifications for the grammar along with documentation
- Sample input program and the expected output.
- The generated output of the assignment.
- Write documentation wherever necessary

READING:

Alfred V. Aho, Ravi Sethi, J.D. Ullman, Compilers, Principles, Techniques, and Tools, Pearson
John Levine, Tony Mason, Doug Brown, Lex & Yacc, O'reilly
John Levine, flex & bison, O'reily-SPD, 2009