CSE321 – Compiler Design (LEX & YACC) Laboratory

Part A – Programs using LEX / YACC

1. LEX program to perform the following [using file]
    a. Recognize the reserved words of C++ and Python separately
    b. replace every occurrence of the word "scanf" to "printf".
    c. recognize the binary string of length atmost 6 and also begins and ends with different symbol
    d. Accept & print the strings that are beginning with only vowel characters.
    e. recognize & count the signed integer and signed floating point number
    f. count the number of lines in the given input.
    g. recognize the different types of operators of C language
    h. recognize an identifier name and user-defined function name in C
    i. stripe out the comment line from the input
    j. Recognize the valid email address.
2. LEX / YACC program to validate the syntactical correctness of an arith_expression
3. LEX / YACC program to convert the Infix to postfix equivalent of the given expression
4. LEX / YACC program to evaluate the resultant value of given arithmetic expression.
5. LEX / YACC program to implement a simple desk calculator
6. LEX / YACC program to validate the syntax of Simple If statement
7. LEX / YACC program to validate the syntax of If-else statement of python
8. LEX / YACC program to validate the syntax of defining a LIST / Dictionary datatype of python.
9. LEX / YACC program to validate the syntax of while loop statement
10. LEX / YACC program to generate the TAC Statement for the given arithmetic expression

Part B – Programs using Python / C / C++

1. Compute FIRST symbols of the given Context Free Grammar [CFG is given]
2. Compute FOLLOW symbols of the given Context Free Grammar {CFG, FIRST are given in the question]
3. Build LL (1) parsing table for the given Grammar (CFG, FIRST and FOLLOW sets are given as input)
4. Perform LL (1) parsing – validating the string (CFG and Parsing table are given as input)
5. Build LR parsing table for the given Grammar. (Context- Free Grammar, Action and GOTO tables are supplied as inputs)
6. Perform SLR parsing - validating the string. (Context- Free Grammar, Action and GOTO tables are supplied as inputs)
7. Code Optimization (TAC statements are given as input)
8. Implement a Local List scheduling algorithm. (TAC sequence will be given)
9. Perform Register allocation and assignment for the given TAC sequence.
10. Build the Back end of the compiler – receives the TAC statement and producing the target code.

Allocation of Marks

Part A			Part B			Viva (10)	Total (50)
Algorithm (5)	Program (10)	Output (5)	Algorithm (5)	Program (10)	Output (5)		

Examiner – 1 **Examiner – 2**