Mid-Semester Examination
School of Computer Engineering
KIIT University, Bhubaneswar-24

Time: 2hrs *Full Mark: 50*

*(Answer any **FIVE** questions including **question no. 1**)*

1. [5 × 2]
   a. Why two buffer halves are used in input buffering?
   b. Name the transformations needed to apply to convert a given grammar into an LL(1).
   c. State the structure of a LEX Program. Write one line for each section of a LEX program.
   d. Remove left recursion from the following Context-Free Grammar.
      S → Aa | Sa| c
      A → Ab | Sd | e
   e. Apply left factoring on the following Context-Free Grammar.
      S → xAy |xBy| xAz
      A → qS | q
      B → q

2. a. Explain the different phases belonging to the front end of a compiler. Show the output of each phase, for the given statement "$x = m / n - p * 30$" (where the variables m, n are of floating point type and x, p are integer type). [06]
   b. Write a LEX program that replaces each instance of the keyword float by double. [04]

3. Consider the following Context-Free Grammar. [10]
      lexp → atom | list
      atom → ***num*** | ***id***
      list → ***(***lexp_seq***)***
      lexp_seq → lexp_seq lexp | lexp

   Consider the string ***(id num(id id id))*** and perform the following:
   a. Give a leftmost derivation for the given string.
   b. Give a rightmost derivation for the given string.
   c. Give a parse tree for the given string.
   d. Is the grammar ambiguous or unambiguous? Justify your answer.

4. a. Construct the LL(1) parsing table for the following grammar: [05]
      S → aAC| Bb
      A → eD
      B → f| g
      C → h | i
      D → bE|ϵ
      E → eD | dD
   b. Write the procedures for the non-terminals of the given grammar to design a recursive-descent parsing without backtracking. [05]
      S → (L)
      L → L, S | a

5. Compute the FIRST and FOLLOW sets for the following grammars. [10]

(a)
S → AB| eDa
A → ab | c
B → dC
C → eC | ε
D → fD | ε

(b)
S → ABBA
A → a | ε
B → b | ε

(c)
S → aAcd | BCe
A → b | ε
B → Cf | d
C → fe

(d)
S → aSe | B
B → bBe | C
C → cCe | d

6. Consider the regular expression **1(1+0)*0**.
Construct the NFA for the given regular expression using Thomson's construction algorithm and then apply subset construction algorithm to construct the equivalent DFA of the constructed NFA. Finally, minimize the DFA. [10]

**********All The Best**********