Reg. No.										

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

VI SEMESTER B.TECH. COMPUTER SCIENCE AND ENGINEERING

MAKE-UP EXAMINATIONS, JUNE/JULY 2018

SUBJECT: COMPILER DESIGN (CSE 3201)

REVISED
13/06/2018

Time: 3 Hours MAX. MARKS: 50

Instructions to Candidates:

- Answer **ALL** the questions.
- Missing data may be suitable assumed.

1A. What are the advantages of: (a) a compiler over an interpreter (b) an interpreter over a compiler? Explain the various modules of language processing system. **3M**

1B. Define token, patterns and lexeme. Divide the following C++ program into appropriate lexemes. Which of the lexemes should get associated lexical values? What should those values be? **3M**

```
float limitedSquare(x) {
float x;
/* returns x-squared, but never more than 100 */
return (x<= -10.0 || x>=10.0)?100 : x*x;
}
```

1C. Explain in detail the various approach used by Lexical Analyzer to handle the reserved words and identifiers. **4M**

2A. Consider the following CFG G = (N={S, A, B, C, D}, T={a,b,c,d}, P, S) where the set of productions P is given below: **4M**

S → A
A → BC | DBC
B → Bb | ε
C → c | ε
D → a | d

Is this grammar suitable to be parsed using the Recursive Descent parsing method? Justify and modify the grammar if needed. Also, construct the corresponding parsing table using the predictive parsing LL method and parsing actions for the input “dbb”

2B. Consider the context-free grammar: **2M**

S -> S S + | S S * | a

and the string aa + a*.

Give the left most and rightmost derivation for the given input string and draw the parse tree and check if the grammar is ambiguous.

2C. For the following grammar, construct the LR(1) DFA, showing all items in each state. And construct the CLR(1) parse table for the same. **4M**

S→ (L) | a
L→ L , S | S

3A. Give the algorithm for computing closure of item set (I). Explain the same using an example. **2M**

3B. Given the grammar below, construct a DFA of LR(0) items. **3M**

(1) Stmts → Stmt
(2) Stmts → Stmts ; Stmt
(3) Stmt → Var = E
(4) Var → id [E]
(5) Var → id
(6) E → id
(7) E → (E)

3C. Explain the two types of semantic attributes and show the propagation of different attributes using the annotated parse tree for the input string 1*2*3*(4+5)n using the grammar given below. **5M**

L→ En
E→TE'
E'→+TE' | ε
T→FT'
T'→*FT' | ε
F→(E) | digit

4A. Generate the assembly level code for the following sequence assuming that x, y, and z are in memory locations. Also, compute the instruction cost of the generated code. **3M**

```
   if x < y goto L1
   z = 0
   goto L2
L1: z = 1
```

4B. Give the TAC representation for the following code segment. And also give the Quadruple representation for the same. **4M**

```
int a[10], b[10], dot_prod, i;
dot_prod=0;
for (i=0; i<10; i++)
dot_prod + = a[i]*b[i];
```

4C. Explain the value number method for constructing DAG's using an example. Construct AST and DAG for the expression: ((x + y) - ((x +y) * (x -y))) + ((x+y) * (x-y)) **3M**

5A. Write a Lex program that converts a file to "Pig latin." Specifically, assume the file is a sequence of words (groups of letters) separated by whitespace. Every time you encounter a word: **3M**

a. If the first letter is a consonant, move it to the end of the word and then add ay!
b. If the first letter is a vowel, just add ay to the end of the word.

5B. Explain the various fields of general Activation record. **3M**

5C. Explain the following characteristics of peephole optimizations with an example for each. **4M**

a. Redundant-instruction elimination
b. Algebraic Simplifications
c. Use of machine idioms

****