Reg. No.										

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL
(A constituent unit of MAHE, Manipal)

VI SEMESTER B.TECH. COMPUTER SCIENCE AND ENGINEERING

END SEMESTER EXAMINATIONS, APRIL/MAY 2019

SUBJECT: COMPILER DESIGN (CSE 3201)

REVISED CREDIT SYSTEM
(25-04-2019)

Time: 3 Hours MAX. MARKS: 50

Instructions to Candidates:

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

1A. What is the role of an assembler in a Language Processing system? Explain the different measures of mapping a source program into a semantically equivalent target program. **3M**

1B. Explain the pointers used in the buffer pairs. Also explain how these pointers in the buffer pair overcome the drawbacks of single input buffer with respect to the following expression. **2M**

pos=i**j

1C. Write the transition diagram for identifying the tokens for hexadecimal and octal constants in C. Also write the pseudo code for implementing the same. (eg for Hex is 0X40L and Oct is 040U) **5M**

2A. Check if the given **Grammar G** is LL (1) by constructing a predictive parse table. Clearly specify the different steps involved during the construction of parse table. **4M**

A→ BCg | DBCe
B→BDb | ε
C→DCf | ε
D→ a | ε
Grammar G

2B. Generate code for the following three-address statements assuming a and b are arrays whose elements are 4-byte values. Also compute the cost involved. **2M**

x = a[i]
y = b[j]
a[i] = y
b[j] = x

2C. Consider the given grammar S→ABC, A → Agd | ε, B → Bd**;** | **,** | ε, C → cC | ε **4M**

a. Compute the canonical set of LR(0) items for the above grammar
b. Build the ACTION/GOTO table
c. Use the ACTION/GOTO table to parse the string "gdgd**,**d**;**c"

3A. Construct an LR (1) Automaton for the given **Grammar A**. Also give the number of states that contain reduce operations. **5M**

S→ pXYZ | ε
X→XSd | ε

Y→Za | hZ | ε
Z→ f | Zg
Grammar A

3B. When do we say that an error has encountered in predictive parsing? With an example, explain the possible ways of selecting a synchronizing token set to recover from an error in panic mode recovery. **3M**

3C. State the important goals of error handler in a parser. Write the pseudo code to eliminate left recursion from a grammar. **2M**

4A. Generate three address code for the following C segment: (consider array elements of 8 bytes). Draw the quadruple for the generated three address code. **5M**

```
s = n*n;
for (i=2; i<=s; i++)
     if (a[i]) {
         count++;
         for( j=2*i; j<=n; j=j+i)
               a[j] = FALSE; }
```

4B. Draw the directed acyclic graph for the expression: **2M**

z=((a+b)*c)+b+((a*a)+b+(a*a))

How many nodes are created for DAG and abstract syntax tree for the above expression? What can you infer from this?

4C. Write an algorithm to partition three address code into basic blocks. Also draw flow graph for the three address code generated in **question 4A**. **3M**

5A. Write a Flex program **3M**

i) To convert the Roman number to Arabic.
ii) To generate tokens for the following grammar.

Query → select Parameters Fclause Wclause
Fclause → from Parameters
Parameters → id | id, Parameters | *
Wclause → where Exp | ε
Exp → id = Number

Note:- Number is a signed double number.

5B. Draw the activation tree for the following program segment. **2M**

```
void Output(int n, int x){
printf("The value of %d! is %d.\n",n,x);
}
int Fact(int n){
int x;
if(n > 1)
x = n * Fact(n-1);
else x=1;
Output(n,x):
return x;
}
void main(){
Fact(4);
}
```

5C. Draw annotated parse tree showing dependency edges for the input string "*3+x-y*" using the **Grammar B** given below to evaluate an expression. Also derive the syntax directed definition for constructing syntax tree. **5M**

E → TE'
E' → +TE' | -TE' | ε
T → (E) | id | num
Grammar B