Reg. No.										

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

A Constituent Institution of Manipal University

VI SEMESTER B.TECH. COMPUTER SCIENCE AND ENGINEERING

END SEMESTER EXAMINATIONS, APRIL 2017

SUBJECT: COMPILER DESIGN (CSE 3201)

REVISED CREDIT SYSTEM
(20-04-2017)

Time: 3 Hours MAX. MARKS: 50

Instructions to Candidates:

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

1A. With a neat diagram, explain the various components of language processing system. **3M**

1B. With the help of an example, explain how is pattern matching done based on NFAs in a Lexical Analyzer Generator? **4M**

1C. Give the transition diagram for accepting and returning tokens for unsigned numbers as per the given regular definition. (**Note**: Use retract function wherever required) **3M**

$$
\begin{aligned}
\textit{digit} &\rightarrow 0 \mid 1 \mid \cdots \mid 9 \\
\textit{digits} &\rightarrow \textit{digit}\ \textit{digit}^{*} \\
\textit{optionalFraction} &\rightarrow .\ \textit{digits} \mid \epsilon \\
\textit{optionalExponent} &\rightarrow (\ \mathtt{E}\ (\ + \mid - \mid \epsilon\ )\ \textit{digits}\ ) \mid \epsilon \\
\textit{number} &\rightarrow \textit{digits}\ \textit{optionalFraction}\ \textit{optionalExponent}
\end{aligned}
$$

2A. Construct a predictive parse table for the given grammar. **2M**
S->ABCDE A->a | ϵ B->b | ϵ C->c D->d | ϵ E->e | ϵ.

2B. With the help of parse trees show that the given grammar is ambiguous for the input string "if E1 then if E2 then S1 else S2".Also give the unambiguous grammar for "if-then-else" statements by using the disambiguating rule. Prove the unambiguity of the resulting grammar using the same input string. **4M**

stmt → **if** *expr* **then** *stmt*
| **if** *expr* **then** *stmt* **else** *stmt*
I **other**

2C. Construct LR(1) automaton for the given grammar **4M**
P→PaQ | Q
Q→QR | R
R→Rb | c | d

3A. Is the grammar suitable for top down parsing? If not, convert it such that it becomes suitable for top down parsing. **2M**
rexpr→rexpr + rterm | rterm
rterm → rterm rfactor I rfactor
rfactor → rfactor * | rprirnary
rprimary → a | b

3B. How can LR (0) automata help with shift-reduce decisions? Give the LR-parsing algorithm. **3M**

3C. Give the annotated parse tree for the expression "(3+4)*(5+6)n" clearly showing the propagation of attributes between the different nodes of the tree. Give the semantic rules for the two productions, T→FT' and T'→*FT' of the grammar. **5M**

(**Note**: Make use of the grammar given below to construct the tree)

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

4A. Generate assembly level code for the following sequence assuming that 'n' is in memory location: **3M**

```
   s = 0
   i = 0
L1: if i > n goto L2
   s = s + i
   i = i + 1
   goto L1
L2:
```

4B. How is DAG different from a syntax tree? Draw DAG for the following expression "z=a * a – 2 * a * b + b * b + a * a + 2 * a * b + b * b". Also give the Three address code for the following C-code segment. **4M**

```
if(a<b && b<c)
 {
   a=5;
 }
else if(a)
 {
   a=100;
 }
else {
   a=a+1;
 }
```

4C. Write the quadruple for the C-code segment given in Q4B. **3M**

5A. Explain the detailed structure of a LEX program with a sample example code. **3M**

5B. Given below is a C code to compute Fibonacci numbers recursively. Suppose that the activation record for f includes the following elements in order: (return value, argument n, local s, local t); there will normally be other elements in the activation record as well. The questions below assume that the initial call is f(5). **3M**

```
int f(int n) {
   int t, s;
   if (n < 2) return 1;
   s = f(n-1);
   t = f(n-2);
   return s+t;
}
```

a. Show the complete activation tree.
b. How does the stack and its activation records look like the first time f(1) is about to return?

5C. Explain the method of peephole optimization. For the three address code generated in Q4B, draw the basic block and the flow graph. **4M**

****