Reg No.									

MANIPAL INSTITUTE OF TECHNOLOGY
(Constituent Institute of Manipal University)
MANIPAL-576104

SIXTH SEMESTER B.E. (CSE) DEGREE END SEMESTER EXAMINATION
MAY 2012
LANGUAGE PROCESSORS (CSE 302)
DATE: 17-05-2012

TIME: 3 HOURS MAX.MARKS: 50

Instructions to Candidates

- Answer **any five** full questions.

1A. For the C assignment statement *a[i+1] = a[i] + 2*, draw a parse tree and a syntax tree for the expression clearly marking all the nodes of the tree. [1+1= 2M]

1B. Give a regular definition for the whitespace in a programming language. Why C programming language is referred to as a free format language? [1+2= 3M]

1C. Give a format of a Lex input file identifying all its parts. Using this format, write a Lex program that will convert all uppercase letters to lowercase, except for letters inside C-style comments. [2+ 3=5M]

2A. Give an example of a grammar to show inessential ambiguity. [2M]

2B. Show that the following attempt to solve the dangling else ambiguity is still ambiguous by drawing two parse trees corresponding to two rightmost derivations for the string *if (0) if (1) other else other*. Show the numbering of internal nodes in each parse tree by the step number of the derivation. What does this numbering denote?

stmt → *if (exp) stmt* | *matched-stmt*
matched-stmt → *if (exp) matched-stmt else stmt* | *other*
exp → *0*|*1* [2+1= 3M]

2C. Given the grammar *G*

stmt → *assign-stmt* | *call-stmt* | *other*
assign-stmt → *identifier:= exp*
call-stmt → *identifier (exp-list)*
other → *if (exp) stmt* | *if (exp) stmt else stmt*
exp → *0*|*1*

Transform the grammar G into EBNF rules and obtain the new grammar $G^{1.}$. Write pseudo code to parse this grammar G^{1} by recursive descent. [1+4 = 5M]

3A. For the following grammar, show the resulting grammar after removing the left recursion.

lexp → *atom* | *list*
atom → *number* | *id*
list → *(lexp-seq)*
lexp-seq → *lexp-seq lexp* | *lexp*

Construct FIRST and FOLLOW sets for the non terminals of the resulting grammar. [1+4 = 5M]

3B. Show that the resulting grammar of Question 3(A) is LL(1). [1M]
3C. Construct the LL(1) parsing table for the resulting grammar of Question 3(A) and show the actions for the input string *((a)2)* [2+2 = 4M]

4A. Show definition section of YACC specification for an ambiguous expression grammar considering the operators {+, - * , /} [1M]
4B. For the following grammar

$$E \rightarrow (L) \mid a$$
$$L \rightarrow EL \mid E$$

i) Using the DFA of LR(0) items, construct LR(0) parsing table. Show the parsing action for the input string *(a a)* [2+1+2 = 5M]
ii) Construct SLR (1) parsing table and show the parsing action for the input string *(a a)* [2+2 = 4M]

5A. Consider the following grammar for declaration of identifiers

D→T L
T→int | float
L→L,id |id

Write the semantic rules for checking the type information. Draw an annotated parse tree for the sentence *float x,y,z* [3M]
5B. Briefly explain

i. Sequential Declaration
ii. Collateral Declaration
iii. Recursive Declaration [3 M]

5C. Consider the following code

```
int x,y;
int func(int u, int v)
{       double  x;
        char y;
        --------
        {   char x;
            ------
        }
        {   double y;

           -----------
        }
}
```

How many blocks are available in the given code and show which variable declarations comes under which block and show the complete symbol table. [4 M]

6A. Translate the expressions (a + b) * (c + d) * (b + c) into Quadruples and Triples. Discuss the comparative advantages and disadvantages of the two representations [4M]
6B. For the instruction x= (x+1) * (x+1), show how the three address code can be optimized by means of a DAG. Clearly show both the initial and optimized three address code. [3M]
6C. How forward references can be handled by the an assembler ? Illustrate with a 8086 code as an example [3M]