Reg No.									

प्रज्ञानं ब्रह्म

Manipal
INSPIRED BY LIFE

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

KNOWLEDGE IS POWER
MANIPAL INSTITUTE OF TECHNOLOGY

FIFTH SEMESTER B.TECH. (CSE) DEGREE MAKE-UP EXAMINATION
DESIGN AND IMPLEMENTATION OF PROGRAMMING LANGUAGES (CSE 301)
DATE: 01-01-2014

TIME: 3 HOURS MAX.MARKS: 50

Instructions to Candidates

- Answer **any five** full questions.

1A. What are the different types of errors in the program found during the translation? Explain with an example for each.

1B. With an example for each in C programming language, explain the different categories of programming language with respect to Structured abstraction.

1C. With an example for each in C++ programming language, explain the different concepts in regularity.

1D. Draw the Syntax diagram for the grammar G=({expr,list}, {a, **,**}, expr, P):

P: expr → **(**list**)** | **a**
list → list**,** expr | expr

(2+2+3+3)

2A. Compute the First and Follow for the grammar G=({bexpr, bterm, bfactor}, {or, and, (,) , true, false}, bexpr, P).

P: bexpr → bexpr **or** bterm | bterm
bterm → bterm **and** bfactor | bfactor
bfactor → **(** bexpr **)** | **true** | **false**

2B. Write the complete program to parse the grammar G=({sentence, nounphrase, verbphrase, article, noun, verb}, {a, the, girl, dog, sees, pets}, sentence, P) using recursive decent parser

P: sentence → nounphrase verbphrase
nounphrase → article noun
article → **a** | **the**
noun → **girl** | **dog**
verbphrase → verb nounphrase
verb → **sees** | **pets**

2C. What is overload resolution? Explain how the overloading is resolved in the following C++ code snippet at each function call.

```
int add(int a, int b){
        return (a+b);
}
double add(double a, double b){
        return (a+b);
}
int add(int a, int b, int c){
        return (a+b+c);
}
```

```
int main()
{
    cout<<add(2, 3);
    cout<<add(2.1, 3.2);
    cout<<add(1, 3, 2);
    cout<<add(2.1, 3);
    return 0;
}
```

(3+3+4)

3A. What will be the environment of the following C code snippet at point 1 and point 2?

```
void func1( ) {
    int a, b;
    func2( );
//point 1
}
```

```
void func2( ) {
    int x, y;
//point 2
}
```

```
void main( ) {
    int i, j;
    func1( );
}
```

3B. Show the symbol table for the following C program at the two points indicated by the comments

a) using lexical scope and
b) using dynamic scope.

What does the program print using each kind of scope rule?

```
#include <stdio.h>
int a,b;
int p(){
    int a, p;
    /* point 1 */
    a = 0; b = 1; p = 2;
    return p;
}
void print(){
    printf("%d\n%d\n",a,b);
}
```

```
void q (){
    int b;
    a = 3; b = 4;
    /* point 2 */
    print();
}
main(){
    a = p();
    q();
}
```

3C. Explain type checking and type inference with respect to C programming language. Give an example for each.

(2+6+2)

4A. With an example for each, explain how the Union and Array type constructors are used in the C language.

4B. What is a sequence operator in C programming language? Identify the evaluation methods used and give the output of the following C code snippet explaining how the final values are assigned to each of the variable.

```
int x=0, z, y=20;
z= x&&y;
x=(y+=z, y=y||x, y+1);
printf("x=%d  y=%d  z=%d",x,y, z);
```

4C. Write the output for the following code correcting the errors if any.

```
public class exam
{
   public static int method1(int w)
   {
      int count = 0;
      while (w!=1)
         if (w % 2 == 0)
         {
            w = w / 2;
            count++;
         }
         else
            w = 3 * w + 1;
      return count;
   }
   public static void main(String[] args)
   {
      System.out.println(method1(10));
      System.out.println(method1(7));
   }
}
```

4D. Write the java program to count the frequency of word appearing in the string accepted from the user.

(2+3+2+3)

5A. Explain the standard evaluation rule for Scheme expressions

5B. Give the box and pointer notation for the Scheme list: **((a ( )) ((c) (d) b) e)**

5C. Explain the different parts of the statement as classified according to First-order predicate calculus.

(2+3+5)

6A. Explain the different granularity levels of processes for parallel execution of the programs with example code snippet for each.

6B. Explain the SIMD and MIMD architecture with a diagram for each.

6C. Explain the Denotational and Axiomatic semantics.

(6+2+2)