ISC SEMESTER 2 EXAMINATION
SAMPLE PAPER - 1
COMPUTER SCIENCE PAPER 1 (THEORY)

Maximum Marks: 35

Time allowed: One and a half hour

*Candidates are allowed an additional **10 minutes** for **only** reading the paper.*

*They must **NOT** start writing during this time.*

---

*Answer **all** questions in **Section A, Section B** and **Section C**.*

While answering questions in Sections A and B, working and reasoning may be indicated briefly.

All working, including rough work, should be done on the same sheet as the rest of the answer.

Section-A

Question 1.

(i) The keyword used by a class to inherit another class is:

(a) import (b) implements (c) extends (d) include

(ii) The property of having multiple functions with same name in a single class:

(a) inheritance (c) data abstraction

(b) overriding (d) overloading

(iii) int Sum(int n)

{ return (n<=0)? 0: n%2 + Sum(n/2); }

With reference to the program code given above, what will the function **Sum( )** return when the value of n=56?

(a) 3 (b) 111000 (c) 11 (d) 111

(iv) Write the statement in Java to replace the word "Jan" with "Febr" from the word "January".

(v) State the principle by which the queue data structure works.

(vi) What is the output of the statement given below?

System.out.print("ELEPHANT".substring(5)+ "ELEPHANT".indexOf('E'));

(vii) Write any one advantage of inheritance.

Section-B

Question 2.

Differentiate between linear recursion and tree recursion.

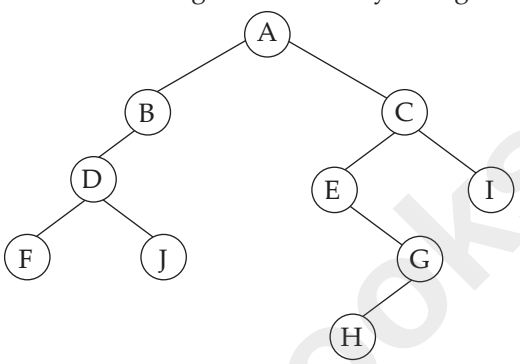
Question 3.

Convert the following infix notation to postfix notation:

A + B * C / D – E + F

Question 4.

Answer the following questions on the diagram of a Binary Tree given below:



(i) State the degree of the nodes C and G. Also, state the level of these nodes when the root is at level 0.

(ii) Write the pre order and post order traversal of the above tree structure.

Section-C

Each program should be written in such a way that it clearly depicts the logic of the problem. This can be achieved by using mnemonic names and comments in the program.

(Flowcharts and Algorithms are not required.)

The programs must be written in Java.

Question 5.

(i) Design a class **CheckWord** which checks whether a word is a palindrome-like word or not. (Palindrome-like words are those which contains the first letter and the last letter same, but rest all letters are different i.e. palindrome-like words are not palindrome itself).

Example: MAGNUM, LOGICAL etc.

The details of the members of the class are given below:

Class name : CheckWord

Data members/instance variables:

wrd : stores a word

len : to store the length of the word

Methods/Member functions:

CheckWord	:	default constructor*/
void acceptWord( )	:	to accept the word in uppercase
boolean palindromeLike( )	:	checks and returns 'true' if the word is a palindrome-like word otherwise returns 'false'
void display( )	:	displays the word along with an appropriate message

Specify the class **Check** giving details of the **constructor, void acceptWord( ), boolean palindrome Like( )** and **void display( )**. Define the **main( )** function to create an object and call the functions accordingly to enable the task.

OR

(ii) Design a class **Alternate** which toggles a word by changing the alternate letters in the word to Uppercase and Lowercase, keeping the first letter unchanged.

For example:

The word "motivate" becomes "mOtIvAtE"

The word "Knowledge" becomes "kNoWlEdGE"

The details of the members of the class are given below:

Class name : Alternate

Data members/instance variables:

str	:	stores a word
newStr	:	stores the changed word
len	:	to store the length of the word

Methods/Member functions:

Alternate	:	default constructor
void readWord( )	:	to accept the word
void change( )	:	converts the alternate letter to Uppercase of lowercase and store it in newStr, keeping the first letter unchanged.
void display( )	:	displays the original word along with the new word.

Specify the class **Toggle** giving details of the **constructor, void readword( ), void alternate( )** and **void display( )**. Define the **main( )** function to create an object and call the functions accordingly to enable the task.

Question 6.

(i) A class **Tribo** has been defined to generate the Tribonacci series 0, 1, 2, 3, 6, 11, 20, ……. (Fibonacci series are those in which the sum of the previous three terms is equal to the next term).

Some of the members of the class are given below:

Class name	**:**	**Tribo**
Data member/instance variable:		
start	:	integer to store the start value
end	:	integer to store the end value
Member functions/methods:		
Tribo( )	:	default constructor
void read( )	:	to accept the numbers
int tribo(int n)	:	return the n^{th} term of a tribonacci series using **recursive technique**
void display( )	:	displays the Tribonacci series from start to end by invoking the function tribo()

Specify the class **Tribo**, giving details of the **Constructor, void read( ), int tribo(int),** and **void display( )**. Define the **main( )** function to create an object and call the functions accordingly to enable the task.

OR

(ii) A class **Lcm** has been defined to find the Lowest Common Multiple of two integer numbers. Some of the members of the class are given below:

Class name	**:**	**Lcm**
Data member/instance variable:		
num1	:	integer to store the first number
num2	:	integer to store the second number
Member functions/methods:		
Lcm( )	:	default constructor
void accept( )	:	to accept the numbers
int lcm(int x,int y)	:	return the LCM of the two number x and y using **recursive technique**
void display( )	:	displays the result with an appropriate message

Specify the class **Lcm**, giving details of the **Constructor, void accept( ), int lcm(int,int),** and **void display( )**. Define the **main()** function to create an object and call the functions accordingly to enable the task.

Question 7.

A class **Demand** contains the market demands of a product. Another class **Supply** is derived from Demand which contains the production and supply details of the product.

The details of the **base class**:

Class Name	**:**	**Demand**
Data member / instance variable:		
PId	:	String to store the product id.
PName	:	String to store the product name.
PDemand	:	Integer to store the demand of the product.
Member functions / methods :		
Demand(…)	:	parameterized constructor to initialize the data members
void display()	:	to display the member data.

The details of the derived class :

Class Name	**:**	**Supply**
Data member /instance variable:		
PProduced	:	to store the amount of product produced.
Prate	:	to store the cost per unit.
Member functions / methods :		
Supply(…)	:	parameterized constructor to initialize the data members of both the classes.
int amount()	:	returns the difference between the amount of demand [rate x demand] and amount of produced [rate x produced].
void display()	:	to display all the details of a product with proper messages (including the difference in amount) (apply method overriding).

Assume that the super class **Demand** has been defined. Using the **concept of inheritance**, specify the class **Supply** giving details of the **constructor, int amount( )** and **void display( )**.

The super class, main function and algorithm need NOT be written.

Question 8.

A Stack is a linear data structure in which the operations are performed based on LIFO (Last In First Out).

Define a class **Stack** with the following details:

Class name	**:**	**Stack**
Data member/instance variable:		
dat[ ]	:	array to hold the integer elements
cap	:	stores the maximum capacity of the stack
top	:	to point the index of the top
Member functions/methods:		
Stack(int max)	:	constructor to initialize the data member cap = max, top = –1 and create the integer array
void push(int v)	:	to add integers at the top index if possible else display the message("Stack full")
int pop( )	:	to remove and return elements from top, if any, else returns – 999
void display( )	:	to display elements of the stack

Specify the class **Stack** giving the details of **void push(int)** and **int pop( )**.

Assume that the other functions have been defined.

The main( ) function and algorithm need NOT be written.

Section-A

Answer 1.

(i) (c) extends

(ii) (d) overloading

(iii) (a) 3

(iv) "January".replace("Jan", Febr");

(v) FIRST IN FIRST OUT (FIFO)

(vi) ANT0

(vii) Code reusability

Answer 2.

Linear recursion	Tree recursion
In this case, the function calls itself only once within its body, thus a linear function call takes place.	In this case, the function calls more than once simultaneously, thus multiple branches of the function call generated.

Answer 3.

= A + (BC*)/D – E + F

= A + (BC*D/) – E + F

= (ABC*D/+) – E +F

= (ABC*D/+E–) +F

= (ABC*D/+E–F+

Answer 4.

(i) when the root is at level 0.

Degree of C = 2

Degree of G = 1

Level of C = 1

Level of G = 3

(ii) Pre order traversal: A B D F J C E G H I

Post order traversal: F J D B H G E I C A

Answer 5.

(i)
```
import java.util.Scanner;
class CheckWord
{
    String wrd;
    int len;

    public CheckWord( )
    {
        wrd="";
        len=0;
    }

    public void acceptword( )
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a word");
        wrd=sc.next( );
        len=wrd.length( );
    }
```

```
    public boolean palindromeLike( )
    {
        if(wrd.charAt(0)!=wrd.charAt(len-1))
            return false;
        for(int i=1, j=len-2; i<=j; i++, j--)
        {
            if(wrd.charAt(i)==wrd.charAt(j))
                return false;
        }
        return true;
    }

    public void display( )
    {
        System.out.print(wrd);
        if(palindromeLike( ))
            System.out.println(" is a Palindrome-like word");
        else
            System.out.println(" is not a Palindrome-like word");
    }

    public static void main(String ar[ ])
    {
        CheckWord ob=new CheckWord( );
        ob.acceptword( );
        ob.display( );
    }
}
```

(ii)

```
import java.util.Scanner;
class Alternate
{
    String str, newstr;
    int len;

    public Alternate( )
    {
        str=newstr="";
        len=0;
    }

    public void readword( )
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a word");
```

```
        str=sc.next( );
        len=str.length( );
    }
    public void change( )
    {
        char c=str.charAt(0);
        newstr+=c;
        if(Character.isUpperCase(c))
        {
            for(int i=1; i<len-1;i+=2)
            {
                char x=str.charAt(i);
                char y=str.charAt(i+1);
                newstr+=Character.toLowerCase(x);
                newstr+=Character.toUpperCase(y);
            }
        }
        else
        {
            for(int i=1; i<len-1;i+=2)
            {
                char x=str.charAt(i);
                char y=str.charAt(i+1);
                newstr+=Character.toUpperCase(x);
                newstr+=Character.toLowerCase(y);
            }
        }
    }

    public void display( )
    {
        System.out.println("Original string:"+str);
        System.out.println("New string:"+newstr);
    }

    public static void main(String ar[ ])
    {
        Alternate Ob=new Alternate( );
        Ob.readword( );
        Ob.change( );
        Ob.display( );
    }
}
```

Answer 6.

(i)
```
import java.util.*;
class Tribo
{
    int start, end;

    public Tribo ( )
    {
        start=end=0;
    }

    public void read( )
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter start and end value");
        start=sc.nextInt( );
        end=sc.nextInt( );
    }

    public int tribo(int a)
    {
        if(a==0 || a==1 || a==2)
            return a;
        else
            return tribo(a–1)+tribo(a–2)+tribo(a–3);
    }

    public void display ( )
    {
        for(int i=start; i<end; i++)
        {
            int p=tribo(i);
            if(p>=start && p<=end)
            System.out.print(tribo(i)+" ");
        }
    }

    public static void main(String ar[ ])
    {
        Tribo Ob=new Tribo( );
        Ob.read( );
        Ob.display( );
    }
}
```

(ii)
```
import java.util.*;
class LCM
{
    int num1, num2;
    public LCM( )
    {
        num1=num2=0;
    }

    public void accept( )
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter two numbers");
        num1=sc.nextInt( );
        num2=sc.nextInt( );
    }

    public int lcm(int n1, int n2)
    {
        int temp,i=2,res;
        if(n1>n2)
            res=n1;
        else
            res=n2;
            temp=res;
            while(res%n1!=0 || res%n2!=0)
            {
                res=temp*i;
                i++;
            }
            return res;
    }

    public void display( )
    {
        System.out.println(lcm(num1, num2));
    }

    public static void main(String ar[ ])
    {
        LCM Ob=new LCM( );
        Ob.accept( );
        Ob.display( );
    }
}
```

Answer 7.

```
class Supply extends Demand
{
    int PProduced;
    double Prate;

    Supply(int id, String nm, int d, int pr, double rt)
    {
        super(id,nm,d)
        PProduced=pr;
        Prate=rt;
    }

    int amount( )
    {
        double d=PDemand*Prate;
        double e=PProduced*Prate;
        return Math.abs(d–e);
    }

    void display( )
    {
        super.display( );
        System.out.println(PProduced);
        System.out.println(Prate);
    }
}
```

Answer 8.

```
import java.util.*;
class Stack
{
    int dat[ ];
    int cap, top;

    public Stack( int max)
    {
        cap=max;
        dat = new int[cap];
        top=–1;
    }

    public void push(int v)
    {
        top++;
        if(top==cap)
```

```
        {
            System.out.println("Queue full");
            top– –;
            return;
        }
        dat[top]=v;
    }

    public int pop( )
    {
        if(top<0)
        {
            System.out.println("Stack empty");
            return –999;
        }
        else
            return dat[top– –];
    }

    public void display( )
    {
        for(int i=top; i>=0; i– –)
        {
            System.out.println(dat[i]);
        }
    }
}
```

❑❑