ISC SEMESTER 2 EXAMINATION
SAMPLE PAPER - 4
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) Recursion:
- (a) is a process of calling one function again and again
- (b) is a process of calling one function within its body
- (c) is a process of calling multiple functions with same name
- (d) is a process of calling one function by different names

(ii) Which of the following statement is true about inheritance?
- (a) It leads to data hiding
- (b) It facilitates code re-usability
- (c) Both (a) and (b)
- (d) none of them

(iii)
```
void print(String s, int a)
{
    if(a>=0)
    {
        System.out.print(S.charAt(a));
        print(s, a-1);
    }
}
```
With reference to the program code given above, what will the function **print**( ) returns when the value of s="INDIA" and n=4 ?

(a) INDIA (b) NDIA (c) AIDNI (d) AIDN

(iv) Write the statement in Java to break the string "SUNDAY" in two parts of equal length.

(v) What is function overriding?

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

System.out.print("Welcome".substring(3);

(vii) Give one point of difference between static and dynamic data structure.

Section-B

Question 2.

Define:

(i) Node in Tree

(ii) Base case in recursion

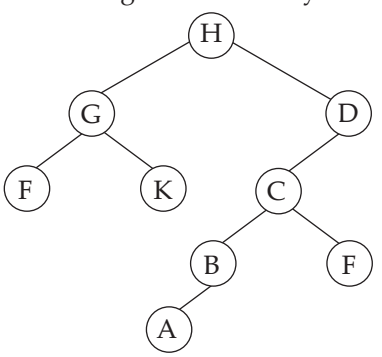
Question 3.

Convert the following infix notation to postfix notation:

P * Q + R ^ S / T

Question 4.

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



(i) Identify the nodes in the longest path of the tree.

(ii) Height and size of the tree when the root is at level 0.

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 **WORDSUM** which will add the ASCII value of all the letters present in the word.

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

Class name	**:**	**WORDSUM**
Data members/instance variables:		
Wrd	:	stores the input word in uppercase
len	:	stores the length of the word
Methods/Member functions:		
WORDSUM( )	:	default constructor
void acceptword( )	:	to accept the word in uppercase
int sumASCII( )	:	add the ASCII values of all the letters and return the same.
void display( )	:	displays the word with the sum

Specify the class **WORDSUM** giving details of the **constructor, void acceptword( ), int sumASCII( )** 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 **Encrypt** which will shift each letter by its position in the word.

Example: TRAIN will become UTDMS

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

Class name	**:**	**Encrypt**
Data members/instance variables:		
Str	:	stores a word in Uppercase
Newstr	:	stores the encrypted word
Len	:	to store the length of the word

Methods/Member functions:

PigLatin( )	:	default constructor
void readword( )	:	to accept the word in uppercase
void encrypt( )	:	shift the letters by its position and store in newstr.
void display( )	:	displays the original word along with the new word

Specify the class **Encrypt** giving details of the **constructor, void readword( ), void encrypt( )** 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 **Series** has been defined to generate sum of the following series:

$$\text{sum} = 1.x + \frac{2.x}{1+2} + \frac{3.x}{1+2+3} + \ldots \text{upto } n \text{ terms}$$

Some of the members of the class are given below:

Class name	**:**	**Series**
Data member/instance variable:		
X	:	integer to store the value of x
N	:	integer to store the value of n (term)
sum	:	double to store the final sum
Member functions/methods:		
Series( )	:	default constructor
void accept( )	:	to accept the value of x and n.
int sum(int a)	:	return the sum of natural numbers up to a using recursion.
void display( )	:	call the sum( ) method and compute the sum of the series.

Specify the class **Series**, giving details of the **Constructor, void accept( ), int sum(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 **Perfect** has been defined to check the number is a perfect number or not.

A perfect number is a number whose sum of the factors excluding itself is equal to the number. For example, 6 = 1+2+3, 28 = 1+2+4+7+14 etc.

Class name	**:**	**Perfect**
Data member/instance variable:		
num	:	integer to store the number
sum	:	integer to store the sum of the factors
Member functions/methods:		
Perfect( )	:	default constructor
void accept( )	:	to accept a positive number.
int factorSum(int a)	:	return the sum of the factors of the given number using recursive technique.
void display( )	:	call the factor() method and check and print the number is palindrome or not and print appropriate message.

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

Question 7.

A super class **DETAIL** has been defined to store the details of a customer. Define a sub class **BILL** to compute the monthly telephone charge of the customer as per the chart given below :

NUMBER OF CALLS	RATE
1 - 100	Only rental charge
101 - 200	₹ 1 per call + rental charge
201 - 300	₹ 2 per call + rental charge
Above 300	₹ 5 per call + rental charge .

The details of both the classes are given below :

Class Name	**:**	**DETAIL**
Data members :		
name	:	To store the name of the customer.
address	:	To store the address of the customer.
telno	:	To store the phone number of the customer.
rent	:	To store the monthly rental charge.
Member functions :		
Detail(..)	:	Parameterized constructor to assign values to data members.
void show()	:	To display the detail of the customer.
Class Name	**:**	**BILL**
Data members :		
n	:	An integer to store the number of calls.
amt	:	A double type data to store the amount to be paid by the customer.
Member functions :		
BILL(…)	:	Parameterized constructor to assign values to data Members of both classes and to initialize amt = 0.0
void calculate()	:	Calculates the monthly telephone charge as per the rate chart given above.
void show()	:	To display the detail of the customer and amount to be paid.

Specify the class **BILL** giving the details of its mentioned methods. Assume that class **DETAIL** is already present.

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

Question 8.

A class **REPEAT** implements a queue using a linear array. Elements can be added at rear position and can be removed from the front position only.

The following details of the class REPEAT are given below :

Class name	**:**	**REPEAT**
Data members		
st[]	:	an array to hold a maximum of 100 integer elements.
capacity	:	stores the capacity of the array.
front	:	to point to the index of the front.
rear	:	to point to the index of the rear.
Member functions		
REPEAT(int m)	:	constructor to initialize the data members capacity = m, front = 0, rear = 0 and to create the integer array.
void pushValue(int v)	:	to add the argument at the rear index if possible else display the message “overflow”, modify rear.

int popValue()	:	to remove and return element from the front, if array is empty then return –9999, modify front.
void reset()	:	reset the array by making front = 0, bringing the elements to the front and modifying rear accordingly.

Specify the class REPEAT giving the details of its **mentioned methods.**

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

Section-A

Answer 1.

(i) (b) is a process of calling one function within its body

(ii) (b) It facilitates code re-usability

(iii) (c) AIDNI

(iv) String a="SUNDAY".substring(0,3);
String b="SUNDAY".substring(3);

(v) Function overriding is two classes, one parent and another child, having functions with same name and same arguments.

(vi) come

(vii) Static data structure has fixed size i.e. it cannot be changed once defined and dynamic data structure can be expanded or shrink as per need

Section-B

Answer 2.

(i) The single unit that have two parts, one as data and another as address.

(ii) It defines the end of recursive call to a function.

Answer 3.

PQ*RS^T/+

Answer 4.

(i) H D C B A

(ii) Height = 5
Size=9

Section-C

Answer 5.

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

    public WordSum( )
```

```
    {
        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 int SumASCII( )
    {
        int s=0;
        for(int i=0; i<len; i++)
        {
            s+=(int)wrd.charAt(i);
        }
        return s;
    }

    public void display( )
    {
        System.out.println(wrd);
        System.out.println(SumASCII( ));
    }

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

(ii)

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

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

```
    public void readword( )
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a word in Upper case");
        str=sc.next( ).toUpperCase( );
        len=str.length( );
    }

    public void encrypt( )
    {
        for(int i=0; i<len;i++)
        {
            char c=str.charAt(i);
            c=(char)(c+(i+1));
            if(c>'Z')
            c-=26;
            newstr+=c;
        }
    }

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

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

Answer 6.

(i)
```
import java.util.*;
class Series2
{
    int x, n;
    double sum;

    public Series2( )
    {
        x=n=0;
        sum=0.0;
    }
```

```
    public void accept( )
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the value of x and n");
        x=sc.nextInt( );
        n=sc.nextInt( );
    }

    public int sum(int a)
    {
        if(a==1)
            return 1;
        else
            return a+sum(a–1);
    }

    public void display ( )
    {
        for(int i=1; i<=n; i++)
        {
            int p=sum(i);
            sum+=(double) i*x/sum(i);
        }
            System.out.print(sum);
    }

    public static void main(String ar[ ])
    {
        Series2 Ob=new Series2( );
        Ob.accept( );
        Ob.display( );
    }
}
```

(ii)

```
import java.util.*;
class Perfect
{
    int num;
    void accept( )
    {
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter a number ");
        num=sc.nextInt( );
    }
```

```
    int factorSum(int a)
    {
        if(a==1)
            return 1;
        else
        {
            if(num%a==0)
            return a+factorSum(a–1);
            else
            return 0+factorSum(a–1);
        }
    }

    void display( )
    {
        if(factorSum(num/2)==num)
        System.out.println("Perfect no");
        else
        System.out.println("Not a perfect no");
    }

    public static void main( )
    {
        Perfect ob=new Perfect( );
        ob.accept( );
        ob.display( );
    }
}
```

Answer 7.

```
    public class Bill extends Detail
    {
        int no; double amt;
        public Bill(String n, String a, int t, int n1)
        {
            super(n, a, t); no = n1;
            amt = 0.0;
        }
        public void Calculate( )
        {
            if(no <= 100)
            amt = rent;
            else if (no <= 200)
            amt = no * 1 + rent;
            else if (no <= 300)
            amt = no * 2 + rent;
        else
            amt = no * 5 + rent;
        }
```

```
    public void show( )
    {
        super.show( );
        System.out.println("The Number of Calls : " + no); System.out.println("The amount to be
        paid : " + amt + "/-");
    }
}
```

Answer 8.

```
import java.util.*;
class REPEAT
{
    int st[ ];
    int capacity, front, rear;

    public REPEAT(int m)
    {
        capacity = m;
        st = new int[capacity];
            front=rear = 0;
    }

    public void pushValue(int v)
    {
        if(rear == capacity)
        System.out.println("Overflow");
        else
        st[rear++] = v;
    }

    int popValue( )
    {
        if (front == rear)
        return – 9999;
        return st[front++];
    }

    public void reset( )
    {
        int i, p = 0;
        for(i = front; i <= rear; i++, p++)
        {
            st[p] = st[i];
            st[i] = 0;
        }
        rear = p;
        front = 0;
    }
}
```

❑❑