ISC SEMESTER 2 EXAMINATION

SAMPLE PAPER - 5

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) What is Recursion in Java?

(a) Recursion is a class.

(b) Recursion is a process of defining a method that calls other methods repeatedly

(c) Recursion is a process of defining a method that calls itself repeatedly

(d) Recursion is a process of defining a method that calls other methods which in turn call again this method

(ii) Which of the following statement is incorrect about String?

(a) String is a class.

(b) Strings in java are mutable.

(c) Every string is an object of class String

(d) Java defines a peer class of String, called StringBuffer, which allows string to be altered

(iii)
```
int compute (int n)
{
    if (n = = 1)
        return 1;
    return fune (n – 1);
}
```
With reference to the program code given above, what will the function compute() returns when the value of n=5?

(a) 0

(b) 1

(c) 120

(d) Error is the code

(iv) Write the statement in Java to swap the first letter with the last letter for the given string "WELCOME".

(v) What is recursive data structure?

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

System.out.print("INDIA".substring(0,2)+ "INDIA".substring(4));

(vii) Give one point of difference between interface and abstract class.

Section-B

Question 2.

Define:

(i) Encapsulation

(ii) Inheritance

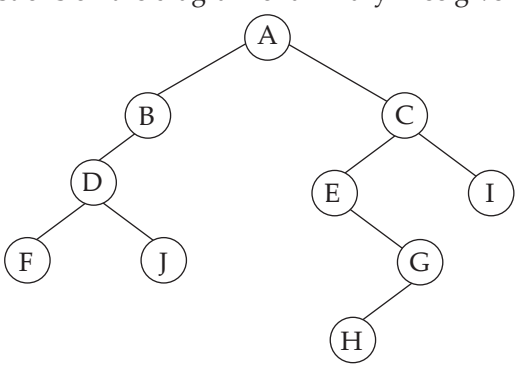
Question 3.

Convert the following infix notation to postfix notation:

(A+B)*(C–D)/E^2

Question 4.

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



(i) Identify internal and external nodes of the tree.

(ii) Write down the post order traversal of the tree.

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 **TWIST** which will SWAP all the letters with the next one present in the word.

For example: RAIN will become ARNI

TODAY will become OTADY

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

Class name	**:**	**TWIST**
Data members/instance variables:		
Wrd	:	stores the input word in uppercase
Res	:	stores the resultant word
Methods/Member functions:		
TWIST( )	:	default constructor
void acceptword( )	:	to accept the word in uppercase
void twist( )	:	swap the letters of the word in Wrd and store the new word in Res
void display( )	:	displays both the words

Specify the class **TWIST** giving details of the **constructor, void acceptword( ), void twist( )** 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 **Palindrome** which will generate a palindrome word from a given word, if that is not a palindrome by adding the missing letters at the end.

Example: DODO will become DODOD

MAD will become MADAM

TOYATA will become TOYATAYOT

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

Class name	**:**	**Palindrome**
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:		
void readword( )	:	to accept the word in uppercase
boolean checkPalindrome(String s)	:	check for s as palindrome or not and return true or false accordingly.
void getPalindrome( )	:	make the palindrome word from the given string Str
void display( )	:	displays the original word along with the new word

Specify the class **Palindrome** giving details of the **void readword(), void checkPalindrome( )** 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 **PrimeSum** has been defined to generate sum of the following series:

sum = 2 + 3 + 5 + 7 + … (add upto n prime numbers)

Some of the members of the class are given below:

Class name	**:**	**PrimeSum**
Data member/instance variable:		
N	:	integer to store the final sum
sum	:	integer to store the value of n (term)
Member functions/methods:		
PrimeSum( )	:	default constructor
void accept( )	:	to accept the value of n.
int prime(int n, int a)	:	return the number (n) is it is prime else return 0 using recursion.
void display( )	:	call the prime() method and add all the prime numbers up to n terms and display the sum.

Specify the class **Series**, giving details of the **Constructor, void accept( ), int prime(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 **BintoDec** has been defined to convert Binary number to its equivalent.

Decimal number.

Class name	**:**	**BintoDec**
Data member/instance variable:		
bin	:	integer to store the decimal
dec	:	integer to store the binary number
Member functions/methods:		
BintoDec( )	:	default constructor
void accept( )	:	to accept a positive binary number.
int bintodec(int a)	:	convert the given binary number to its equivalent decimal number using recursive technique.
void display( )	:	call the bintodec( ) method and print both the binary and decimal number.

Specify the class **BintoDec**, giving details of the **Constructor, void accept( ), int bintodec(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 **Personal** contains employee details and another class **Retire** calculates the employee's Provident Fund and Gratuity. The details of the two classes are given below:

Class name	**:**	**Personal**
Instance variables:		
Name	:	stores the employee name
Pan	:	stores the employee PAN
basic_pay	:	stores the employee basic salary (in decimals)
Member methods:		
Personal(….)	:	parameterized constructor to assign value to data members
void display( )	:	to display the employee details
Class name	:	Retire
Instance variables:		
yrs	:	stores the employee years of service
pf	:	stores the employee provident fund amount (in decimals)
grat	:	stores the employee gratuity amount (in decimals)
Member methods:		
Retire (…….)	:	parameterized constructor to assign value to data members of both the classes.
void provident( )	:	calculates the PF as (2% of the basic pay) * years of service.
void gratuity( )	:	calculates the gratuity as 12 months' salary, if the years of service is ≥ 10 years else the gratuity amount is nil.
void display ( )	:	Display the employee details along with the Provident Fund and gratuity amount.

Specify the class **Personal** giving details of the **constructor** and member functions **void display( )**. Using the **concept of inheritance**, specify the class **Retire** giving details of **constructor**, and the member functions **void provident( ), void gratuity( )** and the **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:		
Str[ ]	:	array to hold the character 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 character array
void push(char v)	:	to add character at the top index if possible else display the message("Stack full")
char pop( )	:	to remove and return elements from top, if any, else returns '#'
void display( )	:	to display elements of the stack

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

Assume that the other functions have been defined.

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

Answers

Section-A

Answer 1.

(i) (c) Recursion is a process of defining a method that calls itself repeatedly

(ii) (b) Strings in java are mutable.

(iii) (b) 1

(iv) "WELCOME".substring("WELCOME".length( )–1)+
"WELCOME".substring(1, "WELCOME".length( )–1)+
"WELCOME".substring(0,1);

(v) A recursive data structure contains references to itself, such as a list or tree.

(vi) INA

(vii) Interface must have all the member methods abstract whereas an abstract class may have some abstract methods and some non-abstract methods.

Answer 2.

(i) Encapsulation – the process of binding data members and member methods in one unit is encapsulation

(ii) Inheritance – the process of sharing the properties by an existing class with newly created classes is inheritance.

Answer 3.

AB+ * CD- / E2^
AB+CD–* / E2^
AB+CD–*E2^/

Answer 4.

(i) Internal nodes: B,C,D,E,G
External nodes: F,J,H,I

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

Answer 5.

(i)
```
import java.util.Scanner;
class TWIST
{
    String wrd,res;

    public TWIST( )
    {
        wrd=res="";
    }

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

```
    public void twist( )
    {
        int len=wrd.length( );
        for(int i=0;i<len–1;i+=2)
        {
            char c=wrd.charAt(i);
            char d=wrd.charAt(i+1);
            res=res+d+c;
        }
        if(len%2!=0)
        res+=wrd.charAt(len–1);
    }

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

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

(ii)

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

    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 boolean checkPalindrome(String s)
        {
            String t="";
            for(int i=0;i<s.length( );i++)
            t=s.charAt(i)+t;
```

```
        return s.equals(t)?true:false;
    }
    public void getPalindrome( )
    {
        String t="", newstr="";
        if(checkPalindrome(str))
        newstr=str;
        else
        {
            for(int i=0;i<len–1;i++)
            {
                t=str.charAt(i)+t;
                if(checkPalindrome(str+t))
                {
                    newstr=str+t;
                    break;
                }
            }
        }
    }
    public void display( )
    {
        System.out.println("Original string:"+str);
        System.out.println("New string:"+newstr);
    }
    public static void main(String ar[ ])
    {
        Palindrome Ob=new Palindrome( );
        Ob.readword( );
        Ob.getPalindrome( );
        Ob.display( );
    }
}
```

Answer 6.

(i)
```
import java.util.*;
class PrimeSum
{
    int n;
    int sum;

    public PrimeSum( )
    {
        n=0;
        sum=0;
    }
```

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

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

int prime(int n, int a)
{
    if(n==1)
        return 0;
    else if(n==2||n==3||n==5||n==7)
        return n;
    else if(n%a==0)
        return 0;
    else if(a>n/2)
        return n;
    else
        return prime(n,a+1);
}

public void display ( )
{
    int p=2, i=1;
    while(i<=n)
    {
        sum+=prime(p,2);
        if(prime(p,2)!=0)
        {
            i++;
        }
        p++;
    }
    System.out.print(sum);
}
```

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

(ii)
```
import java.util.*;
class BintoDec
{
    int bin,dec;
    void accept( )
    {
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter a positve binary number ");
        bin=sc.nextInt( );
        dec=0;
    }

    public int bintodec(int n)
    {
        if(n==0)
            return 0;
        else
            return (n%10) + 2* bintodec(n/10);
    }

    void display( )
    {
        dec=bintodec(bin);
        System.out.println("Binary no."+bin+"\nDecimal no."+dec);
    }

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

Answer 7.

```
class Retire extends Personal
{
    int yrs;
    double pf;
```

```
    double grat;

    public Retire(String n, String p, double b, int y)
    {
        super(n,p,b);
        yrs=y;
        pf=grat=0;
    }

    public void provident( )
    {
        pf=(2.0*basic_pay*yrs)/100.0;
    }

    public void gratuity( )
    {
        if(yrs>=10)
        grat=basic_pay*12;
        else
        grat=0;
    }

    public void display( )
    {
        super.display( );
        System.out.println("PF Amount: "+pf);
        System.out.println("Gratuty: "+grat);
    }
}
```

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(char v)
    {
        top++;
        if(top==cap)
        {
            System.out.println("Stack full");
            top– –;
            return;
        }
        dat[top]=v;
    }

    public char pop( )
    {
        if(top<0)
        {
            System.out.println("Stack empty");
            return '#';
        }
        else
        return Str[top– –];
    }

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

❑❑