KARNATAKA ICSE SCHOOLS ASSOCIATION

ISC STD. XII Preparatory Examination 2023

SUBJECT: Computer Science Paper – I(Answer Key)

Time Allowed: 3 Hours **Maximum Marks: 70** **Date: 19.01.2024**

(Candidates are allowed additional 15 minutes for only reading the paper. They must NOT start writing during this time.)

*Answer **all** questions from **Part-I(compulsory)** and six questions from **Part-II**, choosing **two** questions from Section A, **two** from Section B and **two** from Section C. All working including rough work, should be done on the same sheet as the question. The intended marks for questions or parts of questions are given in brackets [ ]*

PART- I [20 Marks]

*Attempt **all** questions.*
While answering the questions in this Part, briefly indicate the working and reasoning, wherever required.

Question 1

(i) If A and B are the inputs of a half adder, the sum of the half adder is **[1]**

(a) A OR B

(b) A AND B

(c) A XOR B

(d) A EX-NOR

(ii) **Assertion(A)**: Java uses string constant pool and once a String object is created with a value, we are not allowed to perform any changes in that object.

Reason(R) : Strings in Java are immutable. **[1]**

(a) Both Assertion(A) and Reason(R) are true, and Reason is the correct explanation for Assertion.

(b) Both Assertion(A) and Reason(R) are true, but Reason is not the correct explanation for Assertion.

(c) Assertion(A) is true and Reason(R)is false.

(d) Assertion(A) is false and Reason(R) is true.

(iii) All interfaces in Java are _____________ by default. **[1]**

(a) public and private

(b) private and protected.

(c) public and abstract

(d) public and protected.

(iv) **Assertion(A)**: Half adder is faster than a full adder.

Reason(R) : Half adder produces only one output while a full adder gives two outputs. **[1]**

(a) Both Assertion(A) and Reason(R) are true, and Reason is the correct explanation for Assertion.

(b) Both Assertion(A) and Reason(R) are true, but Reason is not the correct explanation for Assertion.

(c) Assertion(A) is true, and Reason(R)is false.

(d) Assertion(A) is false, and Reason(R) is true.

(v) What is the complexity of the following code segment? **[1]**

```
int a=0,b=0;
for(int i =0; i<N;i++)
{
        a *=10;
}
for(int j =0;j<M;j++)
{
        b = b*8 + a;
}
```

(a) O(N*M)

(b) O(N+M)

(c) $O(NM^2)$

(d) $O(N^2 + M^2)$

(vi) Which of these is not a correct statement? **[1]**

(a) Every class containing abstract method must be declared abstract.

(b) Abstract class defines only the structure of the class not its implementation.

(c) Abstract class can be initiated by the new operator.

(d) Abstract class can be inherited.

(vii) Find the complement of the expression P + Q'R. **[1]**

Ans: (P+Q'R)'

= P' . (Q'R)'	Since, De Morgan's Theorem (A+B)' = A' . B'
= P'. ((Q')' + R')	Since, De Morgan's Theorem (A.B)' = A' + B'
= P' . (Q + R')	Involution Law (A')' = A

Therefore, (P + Q'R) = P'Q + P'R' or P' . (Q + R')

(viii) Differentiate between Direct and Indirect recursion. **[1]**

Ans: ***Direct Recursion*** – is a type of recursion where a function calls itself from within itself
Indirect Recursion – is a type of recursion where more than one function call one another mutually.

(ix) What happens if there is no base case in a recursive method? **[1]**

Ans: When no base case is defined in a recursive method, the method calls itself infinitely.

(x) Verify if ((P’ + Q). (Q’ + R))’ + (P’ + R) = 1 **[1]**

Ans: ((P’+Q) . (Q’+R))’ + P’ + R

= **(** (P’+Q)’ + (Q’+R)’ + P’ + R

= (P’)’ . Q’ + (Q’)’ . R’ + P’ + R

= P.Q’ + QR’ + P’ + R

= P’ + PQ’ + R + QR’

= (P+P’) . (P’ + Q’) + (R+Q) (R+R’)

= 1. (P’+Q’) + (R+Q). 1

= P’ + Q’ + R + Q

= P’ + R + Q + Q’

= P’ + R + 1

= P + 1

= 1

Hence Proved.

Or Using Truth Table

P	**Q**	**R**	**(P’+Q)**	**Q’**	**(Q’+R)**	**P’**	**(P’+R)**	**(P’ + Q). (Q’ + R)**	**((P’ + Q). (Q’ + R))’**	**((P’ + Q). (Q’ + R))’ +(P’+R)**
1	1	1	1	0	1	0	1	1	0	1
1	1	0	1	0	0	0	0	0	1	1
1	0	1	1	1	1	0	1	1	0	1
1	0	0	1	1	1	0	0	1	0	1
0	1	1	1	0	1	1	1	1	0	1
0	1	0	1	0	0	1	1	0	1	1
0	0	1	0	1	1	1	1	0	1	1
0	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**

Hence Proved.

Question 2

(i) Convert the following infix notation to its postfix form: **[2]**

A/B - C + D * E – A * C ^ F

Ans: (((A/B) – C) + (D*E)) – ((A * (C^F)))

= ((AB/- C) +(DE*)) – (A*(CF^))

= ((AB/C-)+ DE*) – (ACF^*)

= (AB/C-DE*+) – (ACF^*)

= AB/C-DE*+ACF^*-

Therefore, the postfix expression is AB/C-DE*+ACF^*-

(ii) A Matrix B[10][7] is stored in memory with each element requiring 2 bytes of storage. If the base address at B[x][1] is 1012 and the address at B[7][3] is 1060, determine the value of 'x' where the matrix is stored in column-major orientation. **[2]**

Ans: Given, B[10][7] – Column major-wise storage of elements
No. of Rows(M) = 10
No. of Columns(N) = 7
Size of the elements(W) = 2 bytes
Base Address (B) of B[x][1] = 1012
Address of x(R_0) = ?
Address calculation of a location in an array stored column-major wise is,
Address = B + W [(I – R_0) + M(J – C_0)]
Address of location A[7][3] = 1060
R_0 = ? , C_0 = 1, I = 7 , J = 3
Hence, 1060 = 1012 + 2 [(7 – x) + 10(3 - 1)]

1060 = 1012 + 2[(7-x) + 10 * 2]

1060 – 1012 = 2 [(7-x) + 20]

48 = 2 [(7-x) + 20]

24 = 7 – x + 20

24 – 20 = 7 – x

4 = 7 – x

x = 7 – 4 = 3

Therefore, the value of x(R_0) is 3

(iii) Read the following program segment and answer the questions that follow:

```
void CaseStr(String s, int i)
{
        if(i < s.length())
        {
                char ch = s.charAt(i);
                if(Character.isUpperCase(ch))
```

```
                    System.out.print(Character.toLowerCase(ch));
                else
                    System.out.print(Character.toUpperCase(ch));
                CaseStr(s, i+1);
            }
        }
```

(a) What will be the output of the function CaseStr("MISSISSippi", 0)? **[2]**

Ans: mississIPPI

(b) In one line state what is the task performed by function CaseStr(…) apart from recursion. **[1]**

Ans: The function CaseStr(..) converts the lowercase letters to uppercase letters and vice versa in a given string.

(iv) The following function is a part of the class arrange which stores n integers in an array arr[]. The member/function bubble sort()arranges the array in ascending order.

```
void bubblesort()
{
        int swapped;
        int upto= arr.length;
        do{
                swapped=0;
                for(int i=0;i< ?1?;i++)
                {
                        if(a[i] >arr[i+1])
                        {
                                int val=arr[i];
                                arr[i] = a[i+1];
                                a[i+1]=val;
                                 swapped=1;
                        }
                }
            upto=?2?;
            }while (swapped==?3?);
        }
```

(a) What is the expression/a statement at **?1?** **Ans**: upto – 1 **[1]**

(b) What is the expression/a statement at **?2?** **Ans**: upto = upto --; **[1]**

(c) What is the expression/a statement at **?3?** **Ans**: 1 **[1]**

PART- II [50 Marks]

*Answer **six** questions in this part, choosing **two** questions from Section A, **two** from Section B and **two** from Section C.*

SECTION - A

Answer any ***two*** questions.

Question 3

(i) A company intends to develop a device that shows the high-status power load for a household invertor depending on the criteria given below:

- If the Air conditioner and geyser both are on

  OR
- If the Air conditioner is off, but the Geyser and Refrigerator are on.

  OR
- If the Geyser is off, but Air conditioner and Water purifier are on

  OR
- When all are on

The inputs are:

A	: Air conditioner is on
G	: Geyser is on
R	: Refrigerator is on
W	: Water purifier is on

Output: X [1 indicates a high power, 0 indicates low power for all cases]

Draw the truth table for the inputs and outputs given above and write the SOP expression for X(A,G,R,W). **[5]**

Ans:

A	G	R	W	X	Minterms
1	**1**	**1**	**1**	**1**	**AGRW**
1	**1**	**1**	**0**	**1**	**AGRW’**
1	**1**	**0**	**1**	**1**	**AGR’W**
1	**1**	**0**	**0**	**1**	**AGR’W’**
1	**0**	**1**	**1**	**1**	**AG’RW**
1	0	1	0	0	
1	**0**	**0**	**1**	**1**	**AG’R’W**
1	0	0	0	0	
0	**1**	**1**	**1**	**1**	**A’GRW**
0	**1**	**1**	**0**	**1**	**A’GRW’**
0	1	0	1	0	
0	1	0	0	0	
0	0	1	1	0	
0	0	1	0	0	
0	0	0	1	0	
0	0	0	0	0	

Ans:The SOP Expression is:

AGRW+AGRW’+AGR’W+AGR’W’+AG’RW+AG’R’W’+A’GRW+A’GRW’

X(A,G,R,W) = ∑(6,7,9,11,12,13,14,15)

(ii) Simplify the Boolean expression (A’+C) (A’+C’) (A’+B+C’D) **[3]**

Ans: (A’+C)(A’+C’)(A’+B+C’D)

= (A’A’+ A’C+A’C’+CC’) (A’+B+C’D)

=(0 + A’C+A’C’ +0) (A’+B+C’D)

= A’(C+C’) (A’+B+C’D)

= A’. 1 . (A’+B+C’D)

= A’A’ + A’B + A’C’D

= A’ +A’B+A’C’D

= A’(1+B+C’D)

= A’(1+C’D)

= A’.1

=A’

Therefore, the simplified expression is A’.

(iii) State the best-case and worst-case of a Linear Search algorithm. **[2]**

Ans: Worst case: this can occur when the data item is the last element of the array or not present at all. Hence, f(n) = n=**O(n)**
best case: When the element is in the first position. **f(n) =1**

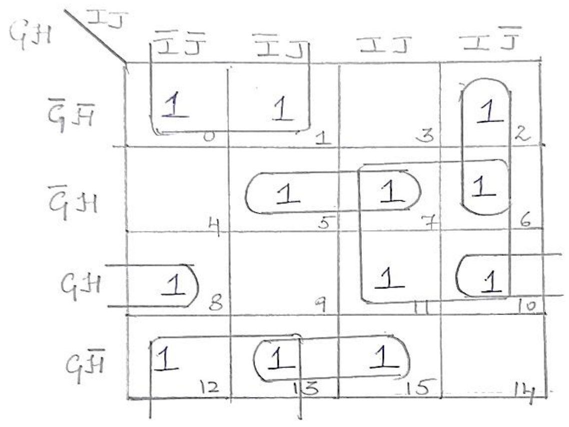
Question 4

(i) Given the Boolean function F(G,H,I,J) =

G’H’I’J’+G’H’I’J+G’H’IJ’+G’HI’J+G’HIJ+G’HIJ’+GHI’J’+GHI’J+GHIJ+GH’I’J’+GH’IJ +GH’IJ’

(a) Reduce the above expression by using a 4-variable Karnaugh map, showing the various groups (i.e. octal, quad or pairs). **[4]**

Ans: Given: F(G,H,I,J) = ∑(0,1,2,5,6,7,8,10,11,12,13,15)



The following groups are obtained from the above K-map,

Quad 1: $m_7 + m_6 + m_{11} + m_{10}$

Quad 2: $m_0 + m_1 + m_{12} + m_{13}$

Pair 1: $m_2 + m_6$

Pair 2: $m_5 + m_7$

Pair 3: $m_8 + m_{10}$

Pair 4: $m_{13} + m_{15}$

The reduced expressions for the above groups are:

Quad 1: HI

Quad 2: H’I’

Pair 1: G’IJ’

Pair 2: G’HJ

Pair 3: GHJ’

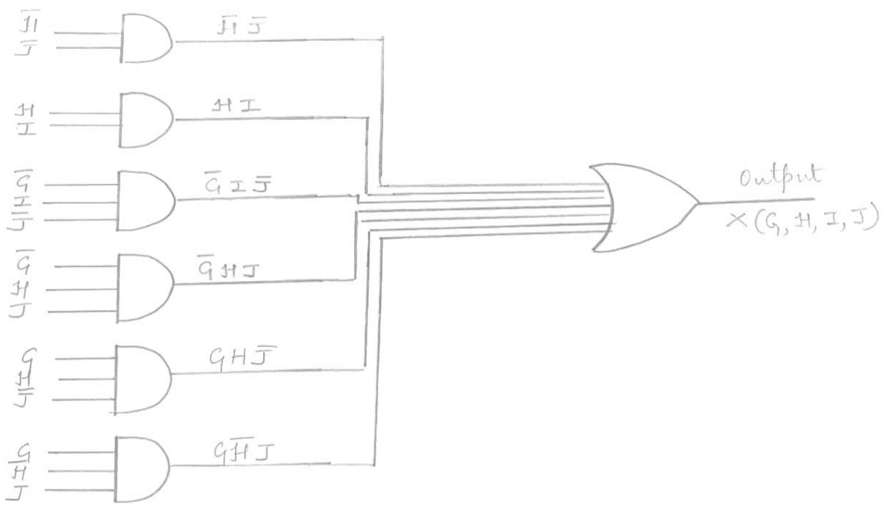
Pair 4: GH’J

Therefore, the reduced SOP expression for the given function is,

HI + H'I' + G'IJ'+ G'HJ + GHJ' + GH'J

(b) Draw the logic gate diagram for the reduced SOP expression. Assume that the variables and their complements are available as inputs. **[1]**

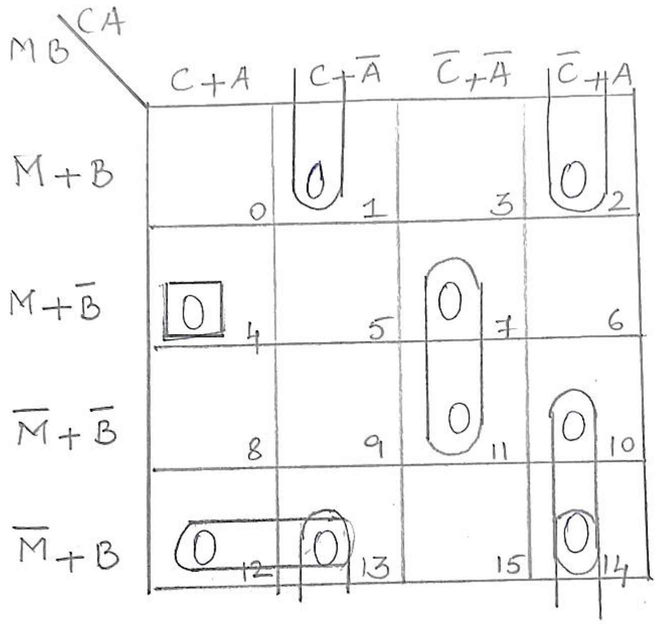
<u>Ans</u>: The logic gate diagram for the above reduced expression is:



(ii) Given the Boolean function X(M,B,C,A) = (M+B+C+A').(M+B+C'+A).(M+B'+C+A).(M+B'+C'+A').(M'+B'+C+A).(M'+B'+C+A').(M'+B'+C'+A). (M'+B+C'+A').(M'+B+C'+A)

(a) Reduce the above expression by using a 4-variable Karnaugh map, showing the various groups (i.e. octal, quad or pairs). **[4]**

<u>Ans</u>: Given: F(M,B,C,A) = π(1,2,4,7,10,11,12,13,14)



The following groups are obtained from the above K-map,

Pair 1: $M_2 . M_{14}$

Pair 2: $M_1 . M_{13}$

Pair 3: $M_{12} . M_{13}$

Pair 4: $M_7 . M_{11}$

Pair 5: $M_{10} . M_{14}$

Individual expression: M_4

The reduced expressions for the above groups are:

Pair 1: B + C’ + A

Pair 2: B + C + A’

Pair 3: M’ + B + C

Pair 4: B’ + C’ + A’

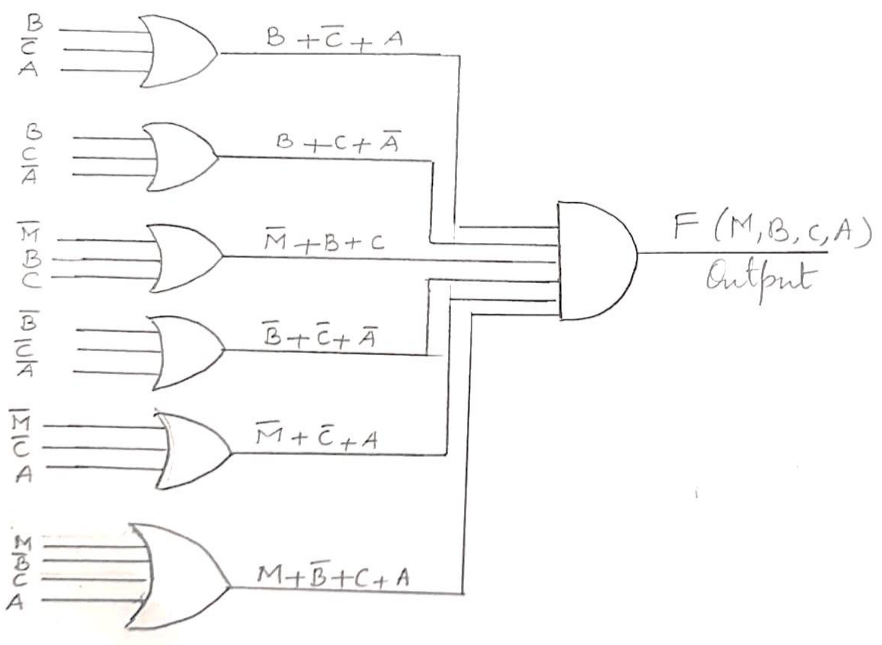
Pair 5: M’ + C’ + A

Therefore, the reduced POS expression for the given function is,

(B+C’+A) . (B+C+A’) . (M’ + B + C) . (B’ + C’ + A’) . (M’ + C’ + A) .(M+B’+C+A)

(b) Draw the logic gate diagram for the reduced POS expression. You may use gates with more than two inputs. Assume that variable and their complements are available as inputs. **[1]**

Ans: The logic gate diagram for the above reduced expression is:



Question 5

(i) State the difference between a multiplexer and a decoder. **[2]**

Ans:

Multiplexer	**Decoder**
A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line	A decoder is a combinational circuit that converts binary information into a maximum of 2^n unique output lines.

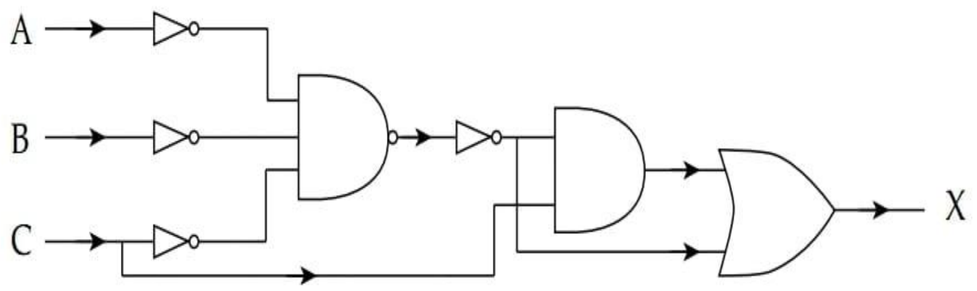
(ii) Verify that (z+x) (z + x' + y) = (z+x) (z+y) **[2]**

Ans:

x	**y**	**z**	**(x+z)**	**x'**	**(z+x'+y)**	**(x+y). (z+x'+y)**	**(z+y)**	**(z+x).(z+y)**
1	1	1	1	0	1	1	1	1
1	1	0	1	0	1	1	1	1
1	0	1	1	0	1	1	1	1
1	0	0	1	0	0	0	0	0
0	1	1	1	1	1	1	1	1
0	1	0	0	1	1	0	1	0
0	0	1	1	1	1	1	1	1
0	0	0	0	1	1	0	0	0

Hence, proved.

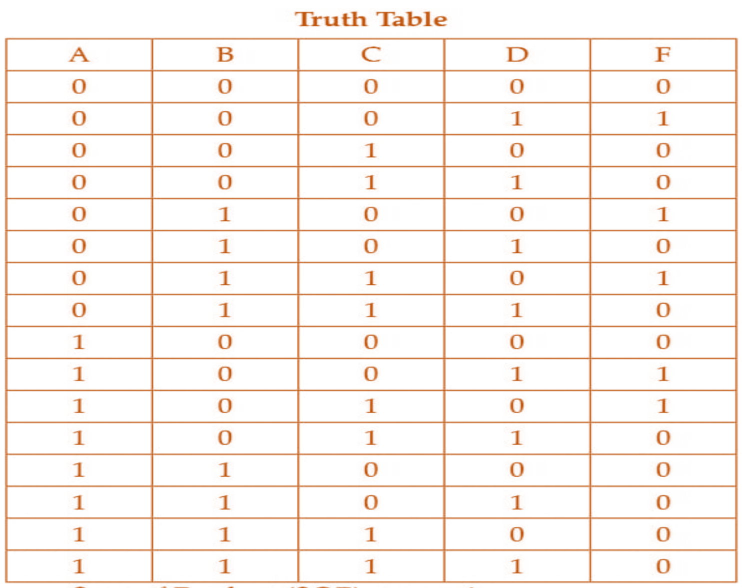
(iii) Derive a Boolean expression for the logic gate diagram given below and reduce the derived expression using Boolean laws. **[3]**



Ans:
The Boolean expression of the above logic diagram is **((A'B'C') . C) + A'B'C'.**
((A'B'C') . C) + A'B'C'
= A'B'C'C + A'B'C'
= 0 + A'B'C' **Since, A.A'=0**
= A'B'C'
Therefore, the reduced expression is A'B'C'.

(iv) Draw the logic diagram and truth table to encode the decimal numbers(1,4,6,9,10) and briefly explain its working. **[3]**

Ans:



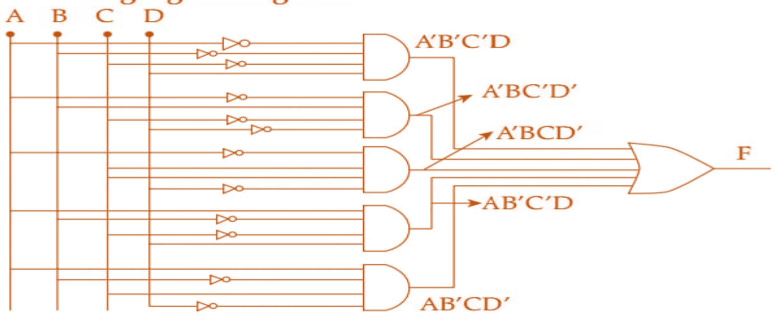
Truth Table

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Sum of Product (SOP) expression

$F(A, B, C, D) = A'B'C'D + A'BC'D' + A'BCD' + AB'C'D + AB'CD'$

Logic gate diagram



SECTION - B

*Answer any **two** questions.*

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 6 **[10]**

A class **SeriesSum** has been defined to compute the following sum of series:

$$\frac{X^2}{1!} + \frac{X^4}{3!} + \frac{X^6}{5!} + \frac{X^8}{7!} + \ldots\ldots\ \frac{X^n}{(n-1)!}$$

Some of the members and functions of the class are as given below:

Class name	: SeriesSum
Data Members	
int x	:to store the value of 'x'.
int n	:to store the limit of the series.
double sum	: to store the sum of the series.
Member methods	
SeriesSum(int x, int nn)	:constructor to initialize the variables.
double findfact(int m)	:to return the factorial of a number 'm' using the Recursive technique.
double power(int a, int b)	:to return 'a' raised to the power 'b' using the Recursive technique.
void calculate()	:to calculate the sum of the series by invoking the recursive functions wherever applicable.
void display()	: displays the sum of the series.

Specify the class **SeriesSum** giving details of the ***constructor(int, int), double findfact(int), power(int, int), void calculate() and void display()***. Define the **main()** method to invoke the functions appropriately to enable the task.

Solution:

```
public class SeriesSum
{
   int x,n;
   double sum;
   public SeriesSum(int x, int nn)
   {
      this.x = x;
      n = nn;
   }

   double findfact(int m)
   {
      double f =1;
      if(m==1 || m==0)
         f = 1;
      else
         f = m * findfact(m-1);

      return f;
   }

   double power(int a, int b)
   {
      double p;
      if(b==0)
         p = 1;
      else
         p = a * power(a,b-1);
      return p;
   }
```

```
  public void calculate()
   {
      for(int i=2;i<=n;i+=2)
      {
         double nr = power(x,i);
         double dr = findfact(i-1);
         sum = sum + nr/dr;
      }
   }

   public void display()
   {
      System.out.println("The sum of the series S =  " +sum);
   }

   public static void main()
   {
      SeriesSum ob = new SeriesSum(2,4);
      ob.calculate();
      ob.display();
   }
}
```

Question 7 **[10]**

Design a class **Combine** to combine the elements of two square matrices in such a way that the lower triangular section of the new matrix is filled with lower triangular elements of the first matrix and its upper triangular section is filled with the upper triangular elements of the second matrix. The left diagonal of the new matrix is the sum of the left diagonal elements of the first and second matrix.

Example:

Matrix A			**Matrix B**			**Matrix C**		
1	2	3	20	21	22	21	21	22
4	5	6	23	24	25	4	29	25
7	8	9	26	27	28	7	8	37

The details of the class are as given below:

Class Name	**: Combine**
Data members	
M[][]	: Array to store integers
n	: stores the order of the matrix. (n x n)
Member functions	
Combine(int x)	: parameterized constructor to initialize the order n = x.
void accept()	: to input the elements in the matrix.

void combineMat(Combine A, Combine B): to fill the matrix of the current object with the elements of the matrices of two parameterized objects A and B according to the above-mentioned process.

void display()   : displays the elements of the matrix.

Define the class **Combine** giving the details of the ***constructor(), void accept(), void combineMat(Combine, Combine) and void display().*** Define a **main()** method to create an object and invoke the functions appropriately.

Solution:

```
import java.util.*;

public class Combine
{
   int M[][] = new int[50][50];
   int n;

   public Combine(int x)
   {
      n = x;
   }
   public void accept()
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the array elements");
      for(int i=0;i<n;i++)
      {
         for(int j = 0;j<n;j++)
         {
            M[i][j] = sc.nextInt();
         }
      }
   }
   public void display()
   {
      System.out.println("The array elements are: ");
      for(int i=0;i<n;i++)
      {
         for(int j = 0;j<n;j++)
         {
            System.out.print(M[i][j] +" ");
         }
         System.out.println();
      }
   }

 public void combineMat(Combine A, Combine B)
   {
      for(int i=0;i<n;i++)
      {
         for(int j=0;j<n;j++)
         {
            if(i==j)
            {
                M[i][j] = A.M[i][j]+B.M[i][j];
            }
            else if(j>i)
            {
                M[i][j] = B.M[i][j];
            }
            else if(j<i)
            {
                M[i][j] = A.M[i][j];
            }
         }
      }
   }

   public static void main()
   {
      Combine A = new Combine(3);
      Combine B = new Combine(3);

      A.accept();
      System.out.println("Matrix A: ");
      A.display();
      B.accept();
      System.out.println("Matrix B: ");
      B.display();

      Combine C = new Combine(3);
      C.combineMat(A,B);
      System.out.println("Matrix C: " );
      C.display();
   }
}
```

Question 8 **[10]**

Encryption is a technique of coding messages to maintain their secrecy. A string array of size ‘n’ where ‘n’ is greater than 1 and less than 10, stores one sentence that ends with a full stop(.), in each row of the array.

Write a program to define a class with the following specifications:

Class name	: Encryption
Data members	
str[ ]	:string array to store a sentence in each location
n	: size of the array

Member methods

Encrpytion(int n)	: parameterized constructor to initialize the size with 'n'
void accept()	: to accept the elements into the string array
void encrypt(String s)	: function which encrypts the odd row sentences such that each character in the sentence is replaced by two characters ahead of it in a circular fashion and displays the encrypted sentence.
void reverse(String s)	: function which stores the even row sentences in reverse of order of its words and displays the reversed sentence.

Specify the class **Encryption** giving details of the ***constructor***(), ***void accept(), void reverse(s), void encrypt(s)***. Define a **main()** function to create an object and call the functions accordingly to enable the task.

Sample Input 1:

N = 4

IT IS CLOUDY

IT MAY RAIN

THE WEATHER IS FINE

IT IS COOL

Sample Output 1:

KV KU ENQWFA

RAIN MAY IT

VJG YGCVJGT KU HKPG

COOL IS IT

Sample Input 2:

N = 13

Sample Output 2: Invalid size entered.

Solution:

```
import java.util.*;
import java.io.*;

public class Encryption
{
   String str[] = new String[50];
   int size;

   public Encryption(int n)
   {
      size = n;
   }
```

```
public void accept()
{
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the strings in the array");
    for(int i=0;i<size;i++)
    {
        str[i] = sc.nextLine();
        if(str[i].charAt(str[i].length()-1) != '.')
        {
            System.out.println("The string must end with a full stop. Re-enter the string");
            i--;
            continue;
        }
    }
}

public void reverse(String s)
{
    String s1="", word="";
    s = s.substring(0,s.indexOf('.'));
    System.out.println(s);
    s = s+' ';
    for(int k=0;k<s.length();k++)
    {
        if(s.charAt(k)!=' ')
            word = word + s.charAt(k);
        else
        {
            s1 = word+' '+s1;
            word ="";
        }
    }
    System.out.println("Reversed String is : " +s1);
}

public void encrypt(String s)
{
    char ch;
    String s1="";

    for(int k=0;k<s.length();k++)
    {
        ch = s.charAt(k);
        if(ch == ' ')
            ch = ch;

        if((ch >=65 && ch<=90) || (ch>=97 && ch<=122))
        {
            ch = (char) (ch+2);
            if(ch>90 && ch<97)
```

```
                {
                ch = (char)((64+ch-90));
                }
                else if(ch >122)
                {
                ch = (char)((96+ch-122));
                }
            }
            s1 = s1+ ch;
        }
              System.out.println("Encrypted String is:  " +s1);
        }

    public static void main()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the array");
        int n = sc.nextInt();
        if(n<1 || n>10)
        {
            System.out.println("Invalid range. Program terminating");
            System.exit(1);
        }
        Encryption ob = new Encryption(n);
        ob.accept();
        for(int i=1;i<=ob.size;i++)
        {
            if(i%2==0)
            {
                ob.reverse(ob.str[i-1]);
            }
            else
            {
                ob.encrypt(ob.str[i-1]);
            }
        }
    }
}
```

SECTION C

Answer any ***two*** questions.

Question 9 **[5]**

A company increases the salary of employees at the beginning of every official year. A super class named **Employee** contains the details of the employee. The sub class **Increment** has been defined to calculate the increment on salary.

The details of the two classes are as follows:

Class Name	**: Employee**
Data members	
String name	: Stores the name of the employee.
String emp_ID	: to store the employee identification number(alphanumeric code).
double sal	: to store the basic salary of the employee.
Member functions	
Employee(String n, String id, int s)	: parameterized constructor to initialize the data members with Parameters.
void show_details()	: displays the details of the employee.
Class Name	**: Increment**
Data members	
int year	: to store the number of years of service
double inc	: stores the increment amount.
double nsal	: stores the new salary after increment.
Member functions	
Increment(String nm, String em, int sa, int y)	: parameterized constructor to initialize the data Members of both the classes and initialize ***inc*** and ***nsal*** with default values.
void calculate()	: calculates the increment and the new salary of the employee based on the following criteria:

Years of service	Increment
Less than 2 years	Nil
2 years to 5 years	Rs. 2000 + 15% of the salary
More than 5 years up to 10 years	Rs. 4000 + 20% of the salary
More than 10 years	Rs. 10000 + 30% of the salary

void show_details()	: displays the details of the employee along with the increment and the new salary.

Assume that the super class **Employee** has been defined. Using the concept of **inheritance**, specify the class **Increment** giving details of the ***constructor(..), void calculate() and void show_details().*** The super class and the **main()** method need not be written.

Solution:

```
public class Increment extends Employee
{
    int year;
    double inc,nsal;
    public Increment(String nm, String em, int sa, int y)
    {
        super(nm,em,sa);
        year = y;
        inc = 0.0;
        nsal = 0.0;
    }

    public void calculate()
    {
        if(year<2)
            inc = 0.0;
        else if(year>=2 && year <=5)
        {
            inc = 2000 + 0.15 * sal;
        }
        else if(year>5 && year <=10)
        {
            inc = 4000 + 0.20 * sal;
        }
        else
        {
            inc = 10000 + 0.30 * sal;
        }
        nsal = sal + inc;
    }

    public void show_details()
    {
        super.show_details();
        System.out.println("Increment :  Rs. " +inc);
        System.out.println("Net Salary:  Rs.  "+nsal);
    }
```

Question 10

Namelist is a linear data structure which works in such a way that the user can enter and remove names from one end only i.e. from the top.

Design a class with the following specifications:

Class Name : Namelist

Data members

List[]	: array to store names
max	: stores the maximum capacity of the array.
top	: to store the index of the top end.

Member methods

Namelist(int m)	: constructor to initialize the data members max = m and top = -1.
void push(String s)	: to add the names to the list at the top location, if possible. Otherwise, display the message "List is full. Cannot add any names further".
String pop()	: Deletes a name from the list at the top location. If there are no names in the List, it returns "####".
void display()	: displays the elements of the list.

(a) Specify the class by giving the details of only the functions **void push(String s) and String pop().** Assume that the other functions have been defined. The main() function need not be written. **[4]**

Solution:

```
public class NameList
{
        int max, top;
        String List[] = new String[50];

        public void push(String s)
        {
                if(top == max-1)
                        System.out.println("List is full. Cannot add any names further");
                else
                        List[++top] = s;
         }

        public String pop()
        {
                String s;
                if(top == -1)
                        return "####";
                else
                {
                        s = List[top];
                        top--;
                        return s;
                 }
            }
```

(b) Name the entity implemented above. What is the principle of the above entity? **[1]**

Ans: STACK, it follows the LIFO(Last In First Out) Principle.

Question 11

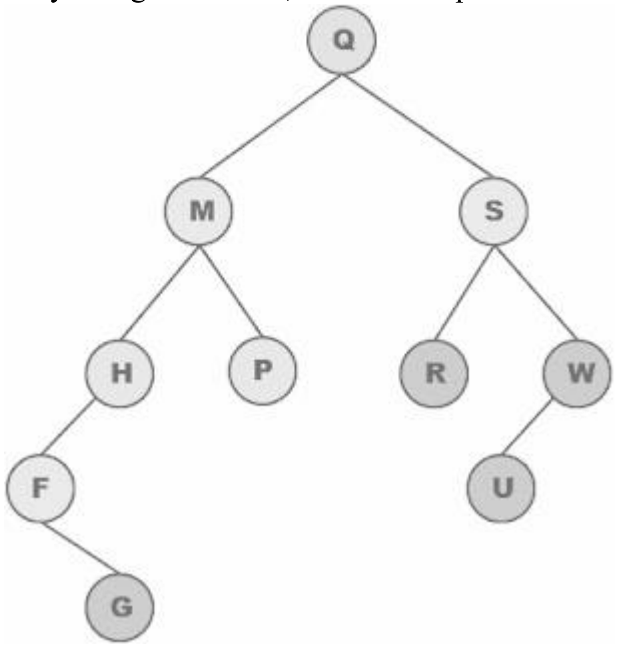
(i) Differentiate between Time complexity and Space complexity. **[1]**

Ans:

Time complexity: It is defined as the number of steps taken to solve an instance of the problem as a function of the size of the input.

Space complexity: It is related to the amount of space/ memory required by an algorithm.

(ii) From the binary tree given below, answer the questions that follow:



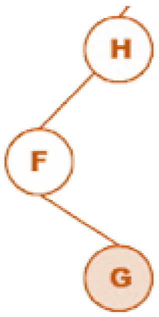
(a) Name the internal nodes of the tree. **Ans**: M, S, H,F, W **[1]**

(b) Write the Pre-order traversal. **Ans:** QMHFGPSRWU **[1]**

(c) What is the height of the node G. **Ans**: 4 **[1]**

(d) Write the sub-tree of node H. **[1]**

Ans:



**********END OF PAPER**********