ISC COMPUTER SCIENCE 2024-2025
PAPER – 1
SOLUTION

Question 1.

(i) The complement of the Boolean expression (A • B') + (B' • C) is: **[1]**

(a) (A + B') • (B' + C)

(b) (A' • B) + (B • C ')

(c) (A' + B) • (B + C ')

(d) (A • B') + (B • C ')

Answer: (c) (A'+B) . (B+C')

Working: $\overline{(A.B')} + (B'.C) = \overline{A.B'} \; . \; \overline{B'.C} \;\; = (A' + B'') \; . \; (B'' + \; C') = (A'+B).(B+C')$

(ii) Given below are two statements marked, Assertion and Reason. Read the two statements carefully and choose the correct option. **[1]**

Assertion: The expression ~ (X ∨ Y) is logically equivalent to (~X ∧ ~Y)

Reason: The commutative property of logical operators states that the order of the operands does not change the result of a binary operation.

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

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

(c) Assertion is true and Reason is false.

(d) Both Assertion and Reason are false.

Answer: (b) Both Assertion and Reason are true, but Reason is not the correct explanation for Assertion

Explanation: (X+Y)' = X' . Y' [by DeMorgan's Law]
X+Y = Y+X [by Commutative Law]
Both are true, but there is no relation between the two laws.

(iii) According to the Principle of Duality, the Boolean equation **[1]**

(1 + Y) • (X + Y) = Y + X' will be equivalent to:

(a) (1 + Y') • (X' + Y') = Y' + X

(b) (0 • Y) + (X • Y) = Y • X'

(c) (0 + Y) • (X + Y) = Y + X'

(d) (1 • Y) + (X • Y) = Y • X'

Answer: (b) (0.Y) + (X.Y) = Y.X'

Explanation: By Principle of Duality – all 1s change to 0s and vice-versa, all ORs change to ANDs and vice-versa, but complement remains unchanged.

(iv) The Associative Law states that: [1]

(a) $A \bullet B = B \bullet A$

(b) $A + B = B + A$

(c) $A \bullet (B + C) = A \bullet B + A \bullet C$

(d) $A + (B + C) = (A + B) + C$

Answer: (d) A + (B+C) = (A+B) + C

Explanation: In Boolean algebra, the associative law refers to the idea that the grouping of variables does not affect the result of the operation. This law applies to both the AND (·) and OR (+) operations.
1. Associative Law for OR: A+(B+C) = (A+B)+C
2. Associative Law for AND: A·(B·C) = (A·B)·C

(v) Consider the following code statement: [1]

```
public class Person
{  int age;
   public Person (int age)
   {
      this.age = age;
   }
}
```

Which of the following statements are valid for the given code?

I. The keyword *this* in the constructor refers to the current instance of the class.

II. The keyword *this* differentiates between the instance variable age and the parameter age.

III. The keyword *this* can be used only in constructors.

(a) Only I and II

(b) Only II and III

(c) Only I and III

(d) Only III

Answer: (a) Only I and II

Explanation: 'this' keyword is used to refer to the current object within a function or a constructor. Hence, only used in constructor is an incorrect statement.

(vi) Given below are two statements marked, Assertion and Reason. Read the two statements carefully and choose the correct option. [1]

Assertion: The break statement prevents fall through effect in switch case construct.

Reason: The break statement enables unnatural exit from the loop.

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

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

(c) Assertion is true and Reason is false.

(d) Both Assertion and Reason are false.

Answer: (b) Both Assertion and Reason are true, but Reason is not the correct explanation for Assertion

Explanation: break can be used in both switch as well as loops, but the purpose of break in switch and loop is not the same. In switch, it is used to prevent fall through where whereas in loop it is used to exit the loop prematurely.

(vii) The canonical expression of $F(P, Q, R) = \pi(2, 5, 7)$ is: [1]

(a) $(P + Q' + R) \cdot (P' + Q + R') \cdot (P' + Q' + R')$

(b) $(P \cdot Q' \cdot R) + (P' \cdot Q \cdot R') + (P' \cdot Q' \cdot R')$

(c) $(P' + Q + R') \cdot (P + Q' + R) \cdot (P + Q + R)$

(d) $(P' \cdot Q \cdot R') + (P \cdot Q' \cdot R) + (P \cdot Q \cdot R)$

Answer: (a) (P+Q'+R) . (P'+Q+R') . (P'+Q'+R')

Explanation: 2 = 010, 5 = 101, 7 = 111 [Binary form of the numbers] Hence, maxterm will be represented by the complement of the literal for every 1 and the literal itself for every 0 in each term.

(viii) Study the given propositions and the statements marked, Assertion and Reason that follow it. Choose the correct option on the basis of your analysis. [1]

P – It is a holiday

Q – It is a Sunday

Assertion: If it is not a Sunday, then it is not a holiday. (Q' => P')

Reason: Inverse is formed when antecedent and consequent are interchanged.

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

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

(c) Assertion is true and Reason is false.

(d) Both Assertion and Reason are false.

Answer: (c) Assertion is true, but Reason is false.

(ix) For the given code segment, write Big O notation for worst case complexity. [1]

```
for ( int i=1; i<=P; i++)
{ Statements }
for (int j=1; j<=P; ++j)
for (int k=1; k<=Q; k++)
{ Statements }
```

Answer: O(P) + O(P.Q) or O(P+P.Q)

Explanation: 1st loop is running P times, hence complexity here is O(P)
2nd loop is a nested loop running PxQ times, hence complexity is O(P.Q)
So, the complexity of the entire code will be – O(P) + O(P.Q)
= O(P+P.Q)
If we consider the dominant term, the complexity will become O(P.Q)

(x) Write the minterms in canonical form for the Boolean Function X (A, B), from the truth table given below: [1]

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Answer: m0 = A'.B' and m3 = A.B OR ∑(0,3)

Explanation: For minterm, we need to consider 1 in the output column i.e. X

Question 2.

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

(A – B / C) + (D * E / F) * G

Answer:
((A – (B/C)) + (((D*E) / F) * G))
(A – BC/) + (((DE*) / F) * G)
(ABC/-) + ((DE*F/) * G)
(ABC/-) + (DE*F/G*)
ABC / - DE * F / G * +

Precedence of operator in priority –
- * / % have same precedence
- + - next in order with the same precedence
- ( ) Overrule all precedence

If two operators have the same precedence, associativity rule applied.

(ii) A matrix M[−1…10, 4….13] is stored in the memory with each element requiring 2 bytes of storage. If the base address is 1200, find the address of M[2][7] when the matrix is stored **Row Major Wise.** [2]

Answer: Formula = B + W * ((I – Lr) * C + (J – Lc))
= 1200 + 2 * ((2+1)*10 + (7-4))
= 1200 + 2 * (30+3)
= 1200 + 2 * 33
= 1200 + 66
= 1266

Total no. of columns, C = 13 – 4 + 1 = 10
Base address, B = 1200
Width of each cell, W = 2
I,J = 2, 7
Lr, Lc = -1, 4

(iii) The following function *int solve( )* is a part of some class. Assume 'm' and 'n' are positive integers. Answer the questions given below with dry run / working.

```
int solve(int m, int n)
{
  int k=1;
  if(m<0)
    return −k;
  else if(m==0)
    return m;
  else
    return k+(solve(m−n, n+2));
}
```

(a) What will the function **solve( )** return if: [2]

(1) **m = 16, n = 1**

(2) **m = 9, n = 1**

(b) What is the function **solve( )** performing apart from recursion? [1]

Answer:

(a) Solve(16,1) ….[m=16, n=1]
= 1 + Solve(15,3) ….[m=15, n=3]
= 1+1 + Solve(12,5) ….[m=12, n=5]
= 1+1+1 + Solve(7,7) ….[m=7, n=7]
= 1+1+1+1 + Solve(0,9) ….[m=0, n=9]
= 1+1+1+1+0
= 4

Solve(9,1)
= 1 + Solve(8,3)
= 1+1 + Solve(5,5)
= 1+1+1 + Solve(0,7)
= 1+1+1+0
= 3

(b) Returning the Square Root of a perfect square.

(iv) The following function **duck( )** is a part of some class which is used to check if a given number is a duck number or not. There are some places in the code marked by **?1?, ?2?, ?3?** which may be replaced by a statement / expression so that the function works properly.

A number is said to be Duck if the digit zero (0) is present in it.

```
boolean duck(int a)
{ int f=−1;
  if(a==0)
    return true;
  for(int i=a; i!=0;?1?)
  {  int c = i%10;
    if(c==?2?)
      {f=1; break; }
  }
  return (f ==?3?)? false:true;
}
```

(a) What is the expression or statement at **?1?** [1]

(b) What is the expression or statement at **?2?** [1]

(c) What is the expression or statement at **?3?** [1]

Answer:

(a) i /= 10 or i = i/10
(b) 0
(c) -1

Question 3.

(i) A superhero is allowed access to a secure Avengers facility if he / she meets any of the following criteria: [5]

- The superhero has Avengers' membership and possesses a high-security clearance badge

OR

- The superhero does not have Avengers membership but holds a special permit issued by S.H.I.E.L.D. along with a high-security clearance badge

OR

- The superhero is not a recognised ally but holds a special permit issued by S.H.I.E.L.D. along with a high-security clearance badge

The inputs are:

INPUTS	
A	Superhero has Avengers membership.
S	Superhero holds a special permit issued by S.H.I.E.L.D.
C	Superhero possesses a high-security clearance badge
L	Superhero is a recognised ally

(In all the above cases, 1 indicates YES and 0 indicates NO)

Output: **X** – Denotes allowed access [1 indicates YES and 0 indicates NO in all cases]

Draw the truth table for the inputs and outputs given above. Write the **POS** expression for **X (A, S, C, L)**.

(ii) Reduce the above expression **X (A, S, C, L)** by using 4-variable Karnaugh map, showing the various groups (i.e., octal, quads and pairs). [5]

Draw the logic gate diagram using **NOR** gates only for the reduced expression. Assume that the variables and their complements are available as inputs.

Answer:

Truth Table

A	S	C	L	X	Maxterm
0	0	0	0	**0**	A+S+C+L
0	0	0	1	**0**	A+S+C+L'
0	0	1	0	**0**	A+S+C'+L
0	0	1	1	**0**	A+S+C'+L'
0	1	0	0	**0**	A+S'+C+L
0	1	0	1	**0**	A+S'+C+L'
0	1	1	0	**1**	
0	1	1	1	**1**	
1	0	0	0	**0**	A'+S+C+L
1	0	0	1	**0**	A'+S+C+L'
1	0	1	0	**1**	
1	0	1	1	**1**	
1	1	0	0	**0**	A'+S'+C+L
1	1	0	1	**0**	A'+S'+C+L'
1	1	1	0	**1**	
1	1	1	1	**1**	

X(A,S,C,L) = (A+S+C+L) . (A+S+C+L') . (A+S+C'+L) . (A+S+C'+L') . (A+S'+C+L) . (A+S'+C+L') . (A'+S+C+L) . (A'+S+C+L') . (A'+S'+C+L) . (A'+S'+C+L')

= ∑(0,1,2,3,4,5,8,9,12,13)

Karnaugh MAP

AS\CL	C+L	C+L'	C'+L'	C'+L
A+S	0	0	0	0
A+S'	0	0		
A'+S'	0	0		
A'+S	0	0		

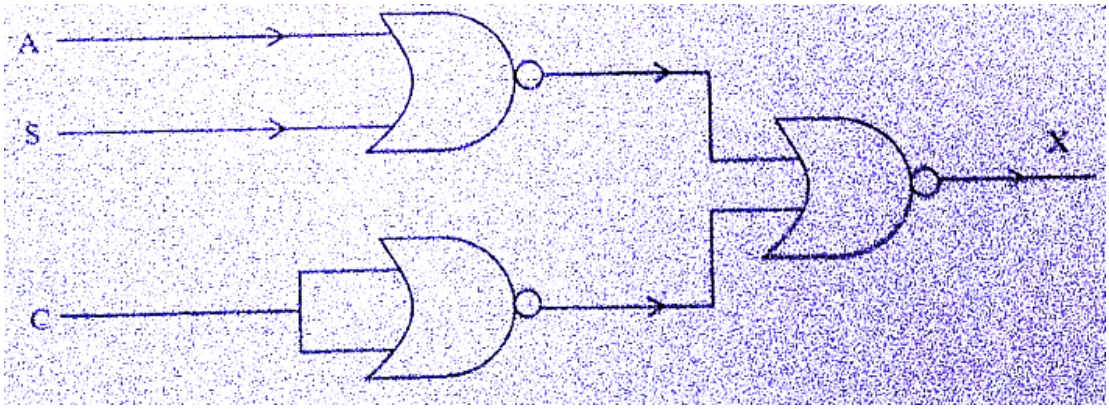
There are 1 Octet and 1 Quad. ***[For easy reference, in the K-Map, the Octet is marked in red color rectangle and the quad is marked in blue color rectangle]***

Octet(m0,m1,m4,m5,m12,m13,m8,m9) = C ***[A, S and L are changing their state, hence removed]***

Quad(m0,m1,m3,m2) = A+S ***[C and L are changing their state, hence removed]***

Reduced expression, X(A,S,C,L) = C . (A+S)

Logic diagram of the reduced expression using NOR gate:



Question 4.

(i) (a) Reduce the Boolean function **F(P, Q, R, S)** = Σ(0,1,2,5,7,8,9,10,13,15) **[4]** by using 4-variable Karnaugh map, showing the various groups (i.e., octal, quads and pairs).

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

Answer:

	R'S'	R'S	RS	RS'
P'Q'	1	1		1
P'Q		1	1	
PQ		1	1	
PQ'	1	1		1

There are 3 Quads.

Quad1 – (m5,m7,m13,m15) = Q.S [AT THE CENTER, IN THICK Blue COLOR]

Quad2 – (m0,m2,m8,m10) = Q'.S' [AT FOUR CORNERS, IN FOUR CIRCLES]

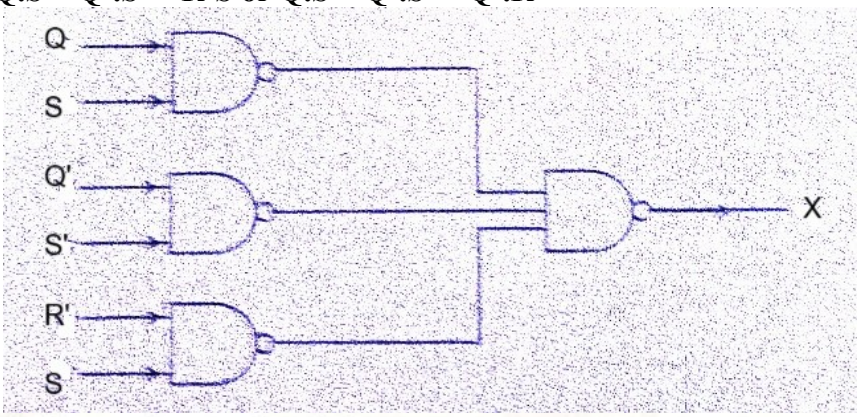
Quad3 – (m1,m5,m13,m9) = R'.S [VERTICAL QUAD AT 2ND COLUMN]

or

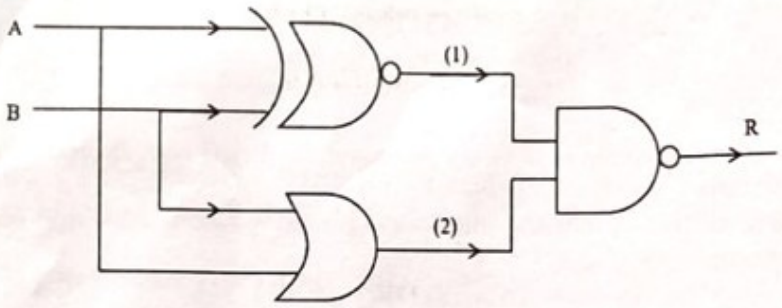
(m0,m1,m8,m9) = Q'.R' [UPPER AND LOWER ROW – 1ST TWO COLUMNS]

Note: Both quads for quad 3 should not be taken as one of them will become a redundant group.

F(P,Q,R,S) = Q.S + Q'.S' + R'S or Q.S + Q'.S' + Q'.R'



(ii) From the logic gate diagram given below:



(a) Derive Boolean expression for (1), (2) and R. Reduce the derived expression. **[4]**

(b) Name the logic gate that represents the reduced expression. **[1]**

Answer:

(a) (1) = (A⊕B)' = A.B + A'.B', (2) = A+B

R = ((A.B + A'.B').(A+B))' = (A.B + A'.B')' + (A+B)'

= (A.B)' . (A'.B')' + A'.B' = (A'+B') . (A+B) +A'B'

= A'.B + A.B' + A'.B'

= A' + A.B' = A'+B' = (A.B)'

(b) NAND gate

Question 5.

(i) What is an *encoder*? Draw the logic gate diagram for an octal to binary encoder. [5]
State *one* application of a *decoder*.

(ii) By using truth table, verify if the following proposition is valid or not. [3]

$(\sim X \Rightarrow Y) \wedge X = (X \wedge \sim Y) \vee (X \wedge Y)$

(iii) Study the logic gate diagram given below and answer the questions that follow:

What will be the output of the above gate when:

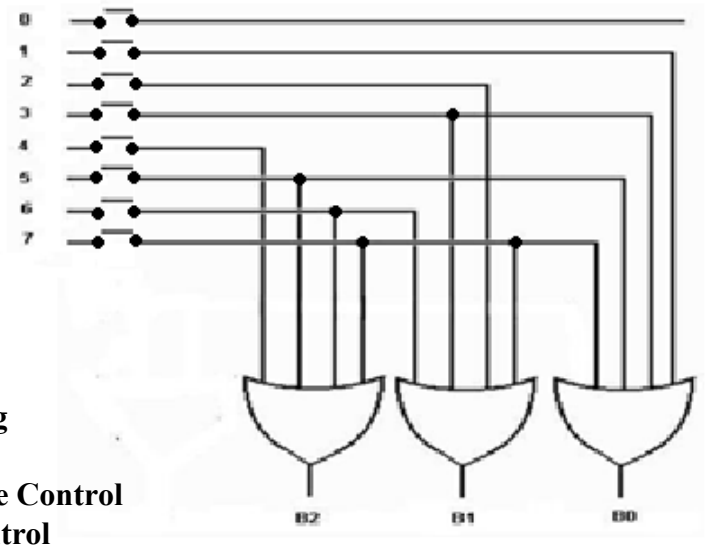
(a) A = 1, B = 0 [1]

(b) A = 1, B = 1 [1]

Answer:

(i) An encoder is a combinational circuit which inputs 2^n or fewer lines and outputs 'n' lines, where 'n' represents the bit code for the input.

Octal to Binary encoder



Application of a decoder

- **Address Decoding**
- **Barcode Scanner**
- **Television Remote Control**
- **Traffic Light Control**
- **Making of Multiplexers**
- **Conversion of Binary to Other Systems (Decimal, Octal, Hexadecimal)**

(ii) Truth table

X	Y	X'	Y'	X'→Y	(X'→Y)^X	X^Y'	X^Y	(X^Y')v(X^Y)
0	0	1	1	0	0	0	0	0
0	1	1	0	1	0	0	0	0
1	0	0	1	1	1	1	0	1
1	1	0	0	1	1	0	1	1

The given expression is VALID.

(iii) (a) 1 or true
(b) 0 or false

Question 6

A class **Perni** has been defined to accept a positive integer in binary number system from the user and display if it is a Pernicious number or not. **[10]**
[A pernicious number is a binary number that has minimum of two digits and has prime number of 1's in it.]
Examples:

- 101 is a pernicious number as the number of 1's in 101 = 2 and 2 is prime number.
- 10110 is a pernicious number as the number of 1's in 10110 = 3 and 3 is prime number.
- 1111 is a **NOT** a pernicious number as the number of 1's in 1111 = 4 and 4 is **NOT** a prime number.

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

Class name	:	**Perni**
Data member/instance variable:		
num	:	to store a binary number
Methods / Member functions:		
Perni( )	:	constructor to initialise the data member with 0
void accept( )	:	to accept a binary number (containing 0's and 1's only)
int countOne(int k)	:	to count and return the number of 1's in 'k' using **recursive technique**
void check( )	:	to check whether the given number is a pernicious number by invoking the function **countOne( )** and to display an appropriate message

Specify the class **Perni** giving the details of the **constructor( ), void accept( ), int countOne(int)** and **void check( )**. Define a **main( )** function to create an object and call the functions accordingly to enable the task.

Answer:

```
import java.util.*;
class Perni
{
   int num;

   public Perni()
   {
      num=0;
   }

   void accept()
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter a positive binary number.");
      int t=Math.abs(sc.nextInt());
      for(int i=t; i>0; i=i/10) //additional code for checking if the input is a binary number or not.
      {
         if(i%10>1) //if any of the digits of num is more than 1, it is no longer a binary number.
         {
            System.out.println("Invalid Binary no.");
            System.exit(0);
         }
```

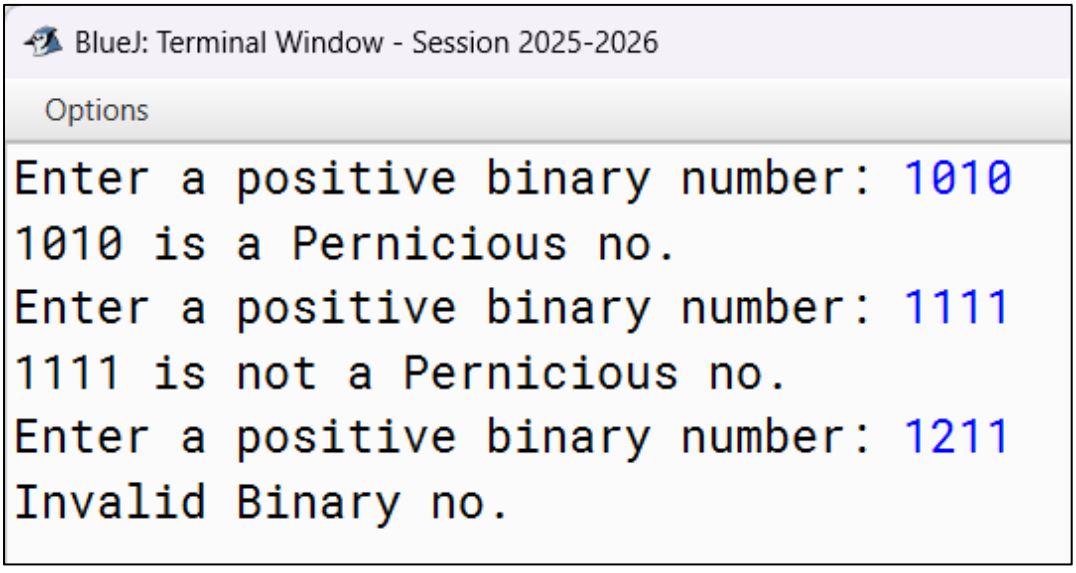
```
        }
        num = t; //storing a valid binary number to num
    }

    int countOne(int k)
    {
        if(k==0) //base case, if the number itself is reduced to 0
        return 0;
        else
        return k%10+countOne(k/10); //adding all the digits of k along with the next recursive call for k/10
    }

    void check()
    {
        int m=countOne(num);
        int k=0;
        for(int i=2;i<=m/2;i++)//loop for checking prime no.
        {
            if(m%i==0)
            {
                k++;//factor other than 1 and m
                break;
            }
        }
        if(k==1||m==0||m==1)
        System.out.println(num+" is not a Pernicious no.");
        else
        System.out.println(num+" is a Pernicious no.");
    }

    public static void main()
    {
        Perni ob=new Perni();
        ob.accept();
        ob.check();
    }
}
```

OUTPUT



Question 7 [10]

Design a class **Colsum** to check if the sum of elements in each corresponding column of two matrices is equal or not. Assume that the two matrices have the same dimensions.
Example:

Input:

MATRIX A

2	3	1
7	5	6
1	4	2

MATRIX B

7	4	2
1	3	1
2	5	6

Output: Sum of corresponding columns is equal.

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

Class name : **Colsum**

Data members/instance variables:

mat[ ] [ ]	:	to store the integer array elements
m	:	to store the number of rows
n	:	to store the number of columns

Member functions/methods:

Colsum(int mm, int nn)	:	parameterised constructor to initialise the data members m = mm and n = nn
void readArray( )	:	to accept the elements into the array
boolean check(Colsum A, Colsum B)	:	to check if the sum of elements in each column of the objects A and B is equal and return true otherwise, return false
void print( )	:	to display the array elements

Specify the class **Colsum** giving details of the **constructor(int, int)**, **void readArray( )**, **boolean check(Colsum, Colsum)**, and **void print( )**. Define the **main()** function to create objects and call the functions accordingly to enable the task.

Answer:

```
import java.util.*;
class Colsum
{
   int mat[][]; //declaring the array without the size
   int m, n;

   public Colsum(int mm, int nn)
   {
      m = Math.abs(mm);
      n = Math.abs(nn);
      mat = new int[m][n]; //memory allocation for the array using new keyword
   }

   void readArray()
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter elements for the matrix");
```

```
    for(int i=0; i<m; i++)
    {
      for(int j=0; j<n; j++)
      {
        mat[i][j]=sc.nextInt();
      }
    }
  }

  boolean check(Colsum A, Colsum B)
  {
    int csumA, csumB;
    for(int j=0; j<A.n; j++) //loop for column elements
    {
      csumA = csumB=0;
      for(int i=0; i<A.m; i++) //loop for rows
      {
        csumA += A.mat[i][j]; //adding column elements of the array from Object A
        csumB += B.mat[i][j]; //adding column elements of the array from Object B
      }
      if(csumA!= csumB) //checking whether both the sums are equal or not equal
      return false;
    }
    return true;
  }

  void print()
  {
    System.out.println("Matrix");
    for(int i=0; i<m; i++)
    {
      for(int j=0; j<n; j++)
      {
        System.out.print(mat[i][j]+"\t");
      }
      System.out.println();
    }
  }

  public static void main()
  {
    Colsum matA= new Colsum(3,4);
    Colsum matB= new Colsum(3,4);
    matA.readArray();
    matB.readArray();
    matA.print();
    matB.print();
    if(matA.check(matA, matB)) //function call with objects passed as parameters to it
    System.out.println("Sum of elements in each column of both matrices is equal");
    else
    System.out.println("Sum of elements in each column of both matrices is NOT equal");
  }
}
```

Question 8 **[10]**

A class **Flipgram** has been defined to flip the letters of the left and right halves of a **non-heterogram** word. If the word has odd number of characters, then the middle letter remains at its own position.
A **heterogram** is a word where no letter appears more than once.

Example 1: INPUT : BETTER
OUTPUT: TERBET
Example 2: INPUT : NEVER
OUTPUT: ERVNE
Example 3: INPUT : THAN
OUTPUT: HETEROGRAM

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

Class name	:	**Flipgram**
Data member/instance variable:		
word	:	to store a word
Methods/Member functions:		
Flipgram(String s)	:	parameterised constructor to assign word = s
boolean ishetero( )	:	to return true if word is a heterogram else return false
String flip( )	:	to interchange the left and right sides of a non-heterogram word and return the resultant word
void display( )	:	to print the flipped word for a non-heterogram word by invoking the method **flip( )**. An appropriate message should be printed for a heterogram word

Specify the class **Flipgram** giving the details of the **constructor(String), boolean ishetero( ), String flip( )** and **void display( )**. Define a **main( )** function to create an object and call the functions accordingly to enable the task.

Answer:

```
import java.util.Scanner;
class Flipgram
{
  String word;
  Flipgram(String s) //parameterized constructor initializing the data member
  {
    word=s.trim();
  }
  boolean ishetero()
  {
    int ctr=0;
    for(int i=0; i<word.length(); i++) //loop for extract each character from the string
    {
      char ch=word.charAt(i);
```

```
        if(word.indexOf(ch)==word.lastIndexOf(ch)) //checking the first occurrence of the letter with the last occurrence
            ctr++; //incrementing the count
        }
        if(ctr==word.length()) //if the count matches with the length of the word
            return true;
        else
            return false;
    }
    String flip()
    {
        int ln=word.length();
        String left=word.substring(0,ln/2); //extracting the left part of the string
        String right="";
        if(ln%2==0) //if the word length is even
            right=word.substring(ln/2); //extracting the right part of the string
        else
        {
            char c=word.charAt(ln/2); //extracting the middle letter
            right=word.substring(ln/2+1); //extracting the right part of the string, excluding the middle letter
            right+=c; //concatenating the right part with the middle letter
        }
        return right+left; //concatenating the right part with the left part of the word
    }
    void display()
    {
        if(ishetero())
        System.out.println("HETEROGRAM");
        else
        System.out.println(flip());
    }
    public static void main()
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a word");
        String s=sc.next();
        Flipgram obj=new Flipgram(s);
        obj.display();
    }
}
```

OUTPUT

```
BlueJ: Terminal Window - Session 2025-2026
Options
Enter a word
BETTER
TERBET
Enter a word
NEVER
ERVNE
Enter a word
SKY
HETEROGRAM
```

Question 9

A circular queue is a linear data structure that allows data insertion at the rear and removal from the front, with the rear end connected to the front end forming a circular arrangement.

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

Class name : **CirQueue**

Data members/instance variables:

Q[ ]	:	array to hold integer values
cap	:	maximum capacity of the circular queue
front	:	to point the index of the front
rear	:	to point the index of the rear

Methods/Member functions:

CirQueue(int n)	:	constructor to initialise cap = n, front = 0 and rear = 0
void push(int v)	:	to add integers from the rear index if possible else display the message "QUEUE IS FULL"
int remove( )	:	to remove and return the integer from front if any, else return −999
void print( )	:	to display the elements of the circular queue in the order of front to rear

(i) Specify the class **CirQueue** giving the details of the functions **void push(int)** and **int remove( )**. Assume that the other functions have been defined. **[4]**

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

(ii) State *one* application of a circular queue. **[1]**

Answer: **(i)**

```
class CirQueue
{
    public void push(int v)
    {   if((rear+1)%cap==front)
            System.out.println("QUEUE IS FULL");
        else
        {   rear=(rear+1)%cap;
            Q[rear]=v;
        }
    }
    public int remove()
    {   if(front==rear)
            return -999;
        else
        {   front=(front+1)%cap;
            return Q[front];
        }
    }
}
```

(ii) Applications of Circular Queue:

- **CPU Scheduling (Round Robin Scheduling)**
- **Memory Management (Buffering)**
- **Network Traffic Management**
- **Printing Spooling**
- **Keyboard Input Buffer**

Question 10 [5]

A superclass **Flight** has been defined to store the details of a flight. Define a subclass **Passenger** to calculate the fare for a passenger.

The details of the members of both the classes are given below:

Class name	:	**Flight**
Data members/instance variables:		
flightno	:	to store the flight number in string
dep_time	:	to store the departure time in string
arr_time	:	to store the arrival time in string
basefare	:	to store the base fare in decimal
Methods/Member functions:		
Flight(…)	:	parameterised constructor to assign values to the data members
void show( )	:	to display the flight details
Class name		**Passenger**
Data members/instance variables:		
id	:	to store the ID of the passenger
name	:	to store the name of the passenger
tax	:	to store the tax to be paid in decimal
tot	:	to store the total amount to be paid in decimal
Methods/Member functions:		
Passenger(…)	:	parameterised constructor to assign values to the data members of both the classes
void cal( )	:	to calculate the tax as 5% of base fare and total amount (base fare + tax)
void show( )	:	to display the flight details along with the passenger details and total amount to be paid

*Assume that the super class **Flight** has been defined*. Using the **concepts of Inheritance**, specify the class **Passenger** giving the details of **constructor(…)**, **void cal( )** and **void show( )**.

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

Answer:

*//code for the child class **Passenger** inheriting the base class **Flight**.*

```
class Passenger extends Flight
{
        String id, name;
        double tax, tot;

        Passenger(String a, String b, String c, double d, String e, String f)
        {
                super(a,b,c,d); //invoking the base class constructor and passing the values
                id=e;   //initializing the child class data members after the base class data members
                name=f;
                tax=tot=0.0;
        }
```

```
    void cal()
    {
        tax=5.0/100.0 * basefare;
        tot = basefare+tax;
    }

    void show()
    {
        super.show(); //invoking base class member method
        System.out.println("ID:"+id);
        System.out.println("Name:"+name);
        System.out.println("Tax:"+tax);
        System.out.println("Total Amount:"+tot);
    }
}
```

Question 11

(i) A linked list is formed from the objects of the class **Cell**. The class structure of the Cell is given below: **[2]**

```
class Cell
{
    char m;
    Cell right;
}
```

Write an *Algorithm* **OR** a *Method* to print the sum of the ASCII values of the lower case alphabets present in the linked list.

The method declaration is as follows:

void lowercase(Cell str)

Answer:

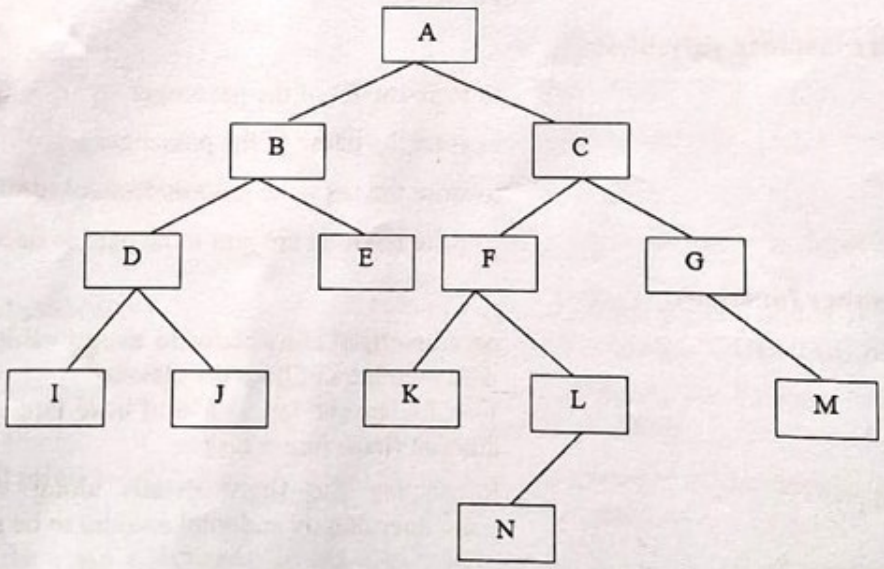
ALGORITHM:

Step 1. Start
Step 2. Set a temporary pointer to the first node.
Step 3. Repeat steps 4 and 5 until the pointer reaches null.
Step 4. Check for lowercase, if found, add ASCII value to sum.
Step 5. Move the pointer to the next node.
Step 6. Display sum.
Step 7. Stop

METHOD:

```
void lowercase(Cell str)
{
    int sum=0;
    Cell T=str;
    while(T!=null)
    {
        if(Character.isLowerCase(T.m))
            sum+=(int)T.m;
        T=T.right;
    }
    System.out.println(sum);
}
```

(ii) Answer the following questions based on the Binary Tree given below:



(a) Write the *in-order* traversal of the right subtree. [1]

(b) State the depth of the entire binary tree and depth of node E. [1]

(c) Name the external nodes of the left subtree and internal nodes of the right subtree. [1]

Answer:

(a) K F N L C G M ***[Inorder traversal is Left-Root-Right traversal]***

(b) Depth of the tree: 4 ***[Depth of a tree – total no. of edges in the longest path]***

Depth of Node E: 2 ***[Depth of a node – total no. of edges from the root node]***

(c) External nodes of left subtree: E I J ***[All the leaf nodes in the left subtree]***

Internal nodes of right subtree: F G L ***[All the parent nodes except the root node]***

-----Thank You-----