ISC EXAMINATION 2023
COMPUTER SCIENCE
(THEORY)
Solved Paper
Class-12th

Maximum Marks: 70

Time allowed: Three hours

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

Answer all questions in 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 rest of the answer.

The intended marks for questions or parts of questions are given in brackets [].

PART I – 20 MARKS

Answer all questions.

While answering questions in this Part, indicate briefly your working and reasoning, wherever required.

Question 1

(i) According to De Morgan's law $(a + b + c')'$ will be equal to: **1**

(a) $a' + b' + c'$ **(b)** $a' + b' + c$

(c) $a' . b' . c'$ **(d)** $a' . b' . c$

(ii) The dual of (X′ + 1) . (Y′ + 0) = Y′ is: **1**

(a) X . 0 + Y . 1 = Y **(b)** X′ . 1 + Y′ . 0 = Y′

(c) X′ . 0 + Y′ . 1 = Y′ **(d)** (X′ + 0) + (Y′ + 1) = Y′

(iii) The reduced expression of the Boolean function F(P, Q) = P′ + PQ is: **1**

(a) P′ + Q **(b)** P

(c) P′ **(d)** P + Q

(iv) If $(\sim p => \sim q)$ then its contra positive will be: **1**

(a) $p => q$ **(b)** $q => p$

(c) $\sim q => p$ **(d)** $\sim p => q$

(v) The keyword that allows multi-level inheritance in Java programming is: **1**

(a) implements **(b)** super

(c) extends **(d)** this

(vi) Write the minterm of F(A, B, C, D) when A = 1, B = 0, C = 0 and D = 1. **1**

(vii) Verify if (A + A′)′ is a Tautology, Contradiction, or a Contingency. **1**

(viii) State *any one* purpose of using the keyboard *this* in Java programming. **1**

(ix) Mention *any two* properties of the data members of an *Interface*. **1**

(x) What is the importance of the *reference* part in a Linked List? **1**

Question 2

(i) Convert the following infix notation to prefix notation. **2**

(A – B)/C*(D + E)

(ii) A matrix M[–6...10m 4...15] is stored in the memory with each element requiring 4 bytes of storage. If the base address is 1025, find the address of M[4][8] when the matrix is stored in **Column Major Wise**. **2**

(iii) With reference to the code given below, answer the questions that follow along with dry run/working.

boolean num(int *x*)

```
{int a=1;
   for(int c=x; c>0’c/=10)
      a*=10;
   return(x*x%a)=x;}
```

(a) What will the function num() return when the value of $x = 25$? **2**

(b) What is the method num() performing? **1**

(iv) The following function task() is a part of some class. Assume '*m*' and '*n*' are positive integers, greater than 0. Answer the questions given below along with dry run/working.

```
  int task(int m, int n)
{if(m==n)
  return m;
else if(m>n)
  return task(m-n,n);
else
  return task(m,n-m);
}
```

(a) What will the function task() return when the value of $m = 30$ and $n = 45$? **2**

(b) What function does task() perform, apart from recursion? **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) Given the Boolean function F(A, B, C, D) = Σ(2, 3, 6, 7, 8, 10, 12, 14, 15).

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

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

(ii) Given the Boolean function F(A, B, C, D) = π(0, 1, 2, 4, 5, 8, 10, 11, 14, 15).

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

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

Question 4

(i) A shopping mall allows customers to shop using cash or credit card or any nationalised bank. It awards' bonus points to their customers on the basis of criteria given below: **5**

- The customer is an employee of the shopping mall and makes the payment using a credit card.

OR

- The customer shops items which carry bonus points and makes the payment using a credit card with a shopping amount of less than ₹10,000

OR

- The customer is not an employee of the shopping mall and makes the payment not through a credit card but in cash for the shopping amount above ₹10,000

The inputs are:

INPUTS	
C	Payment through a credit card
A	Shopping amount is above ₹10,000
E	The customer is an employee of the shopping mall
I	Item carries a bonus point

(In all the above cases, I indicates yes and 0 indicated nd.)

Output: X[1 indicates bonus point awarded, 0 indicates bonus point not awarded for all cases]

Draw the truth table for the inputs and outputs given above and write the **POS** expression for **X(C, A, E, I)**.

(ii) Differentiate between *half adder* and *full adder*. Write the Boolean expression and draw the logic circuit diagram for the SUM and CARRY of a full adder. **3**

(iii) Verify the following expression by using the truth table: **2**

$(A \oplus B)' = (A \oplus B)$

Question 5

(i) What is an *encoder*? How is it different from a decoder? Draw the logic circuit for a 4 : 1 multiplexer and explain its working. **5**

(ii) Form the logic diagram given below, write the Boolean expression for (1) and (2). Also, derive the Boolean expression (F) and simplify it. **3**

(iii) Convert the following cardinal expression to its canonical form: **2**

$F(P, Q, R) = \pi(0, 1, 3, 4)$

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.)

Question 6

Design a class NumDude to check if a given number is a Dudeney number or not. **10**

(A Dudeney number is a positive integer that is a perfect cube, such that the sum of its digits is equal to the cube root of the number.)

Example: $5832 = (5 + 8 + 3 + 2)^3 = (18)^3 = 5832$

Some of the members of the class are given below:

Class name : **NumDude**

Data member/instance/variable:

num : to store a positive integer number

Methods/Member functions:

NumDude() : default constructor to initialise the data member with legal initial value

void input() : to accept a positive integer number

int sumDigits(int *x*) : returns the sum of the digits of number '*x*' using **recursive technique**

void is Dude() : checks whether the given number is a Dudeney number by invoking the function *sumDigits()* and displays the result with an appropriate message

Specify the class **NumDude** giving details of the **constructor(), void input(), int sumDigits(int)** and **void is Dude()**. Define a **main()** function to create an object and call the functions accordingly to enable the task.

Question 7 **10**

A class **Trans** is defined to find the transpose of a square matrix. A transpose of a matrix is obtained by interchanging the elements of the rows and columns.

Example: If size of the matrix = 3, then

ORIGINAL		
11	5	7
8	13	9
1	6	20

TRANSPOSE		
11	8	1
5	13	6
7	9	20

Some of the members of the class are given below:

Class name : **Trans**

Data members/instance variable:

arr[][] : to store integers in the matrix

m : integer to store the size of the matrix

Methods/Member functions:

Trans(int mm) : parameterised constructor to initialise the data member m = mm

void fillarray()	:	to enter integer elements in the matrix
void transpose()	:	to create the transpose of the given matrix
void display()	:	displays the original matrix and the transport matrix by invoking the method *transpose()*

Specify the class **Trans** giving details of the **constructor(), void fillarray(), void transpose()** and **void display()**. Define a **main()** function to create an object and call the functions accordingly to enable the task.

Question 8 **10**

A class **SortAlpha** has been defined to sort the words in the sentence in alphabetical order.

Example: Input: THE SKY IS BLUE

Output: BLUE IS SKY THE

Some of the members of the class are given below:

Class name	**:**	**SortAlpha**
Data members/instance variable:		
sent	:	to store a sentence
n	:	integer to store the number of words in a sentence
Methods/Member functions:		
SortAlpha()	:	default constructor to initialise data members with legal initial value
void acceptsent()	:	to accept a sentence in UPPER CASE
void sort(SortAlpha P)	:	sorts the words of the sentence of object P in alphabetical order and stores the sorted sentence in the current object
void display()	:	display the original sentence along with the sorted sentence by invoking the method *sort()*

Specify the class **SortAlpha** giving details of the **constructor(), void acceptsent(), void sort(SortAlpha)** and **void display()**. Define a **main()** function to create an object and call the functions accordingly to enable the task.

SECTION - C

Answer any two questions.

Each program should be written in such a way that it clearly depicts the logic of the problem stepwise. This can be achieved by using comments in the program and mnemonic names or pseudo codes for algorithms. The programs must be written in Java and the algorithms must be written in general/standard form, wherever required/ specified.

(Flowcharts are not required.)

Question 9

A double ended queue is a linear data structure which enables the user to add and remove integers from either ends i.e., from front or rear.

The details of the class deQueue are given below:

Class name	**:**	**deQueue**
Data members/instance variable:		
Qrr[]	:	array to hold integer elements
lim	:	maximum capacity of the dequeue
front	:	to point the index of the front end
rear	:	to point the index of the rear end
Methods/Member functions:		
deQueue(int I)	:	constructor to initialise lim = 1, front = 0 and rear = 0
void add Front(int v)	:	to add integers in the dequeue at the front end if possible, otherwise display the message "OVERFLOW FROM FRONT"
void add Rear(int v)	:	to add integers in the dequeue at the rear end if possible, otherwise display the message "OVERFLOW FROM REAR"
int popFront()	:	removes and returns the integers from the front end of the dequeue if any, else returns-999
int popRear()	:	removes and returns the integers from the rear end of the dequeue if any, else returns-999
void show()	:	displays the elements of the dequeue

(i) Specify the class **deQueue** giving details of the functions **void addFront(int)** and **int popFront()**. Assume that the other functions have been defined. **4**

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

(ii) Differentiate between a *stack* and a *queue*. **1**

Question 10

A super class **Demand** has been defined to store the details of the demands for a product. Define a subclass **Supply** which contains the production and supply details of the products. **5**

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

Class name	:	**Demand**
Data members/instance variable:		
pid	:	string to store the product ID
pname	:	string to store the product name
pdemand	:	integer to store the quantity demanded for the product
Methods/Member functions:		
Demand(...)	:	parameterised constructor to assign values to the data members
void display()	:	to display the details of the product
Class name		**Supply**
Data members/instance variables:		
pproduced	:	integer to store the quantity of the product produced
prate	:	to store the cost per unit of the product in decimal
Mathods/Member function:		
Supply(...)	:	parameterised constructor to assign values to the data members of both the classes
double calculation()	:	returns the difference between the amount of demand (rate × demand) and the amount produced (rate × produced)
void display()	:	to display the details of the product and the difference in amount of demand and amount of supply by invoking the method *calculation()*

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

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

Question 11

(i) A linked list is formed from the objects of the class given below: **2**

Class Node

```
{
    double sal;
    Node next;
}
```

Write and Algorithm **OR** a *Method* to add a node at the end of an existing linked list. The method declaration is as follows:

void addNode(Node ptr, double ss)

(ii) Answer the following questions from the diagram of a Binary Tree given below:

(a) Write the pre-order traversal of the above tree structure. **1**

(b) Name the parent of the nodes D and B. **1**

(c) State the level of nodes E and F when the root is at level 0. **1**

ANSWERS

1. (i) Option (d) is correct

Explanation: $a'.b'.c$ (Using De Mornan's Law of (A+B)′ = A′ . B′)

(ii) Option (c) is correct

Explanation: X′.0 + Y′.1 = Y′ (Laws of duality is we can obtain another boolean expression by replacing each OR (+) sign with a AND (.) and each AND (.) with OR (+), each 1 with 0 and vice versa.)

(iii) Option (a) is correct

Explanation: P′ + PQ = (P′ + P)(P′ + Q) = 1.(P′ + Q) = P′ + Q)

(iv) Option (b) is correct

Explanation: $q \Rightarrow p$ (A new conditional statement can obtained whose antecedent is the negation of the consequent of the original conditional statement and consequent as negation antecedent of the original conditional statement)

(v) Option (c) is correct

Explanation: Using extends a class we can achieve multilevel inheritance

(vi) A.B′.C′.D (For find any minterm we have to find the SOP expression for variables. Here A=1, B=0, C=0, D=1 so to find the minterm we have to change 0 to 1. To achieve this use the complements of B and C)

(vii) (A + A′)′ = A′.A′′ = A′.A = 0 Hence, it is a contradiction. Following are the truth table of the above expression:

A	A′	A + A′	(A + A′)′
0	1	1	0
1	0	1	0

(viii) **this** keyword returns the address of the current object during the execution.

(ix) **(a)** Data members of an Interface are always declared with **final** keyword.

(b) Data members of an Interface cannot be changed in the due course of the program.

(x) Reference part in a Linked List holds the address of the next node in the Linked List.

2. (i) Prefix of the (A–B)/C*(D+E) :

```
  (A-B)/C*(D+E)
=(-AB)/C*(+DE)
=(/-ABC)*(+DE)
=*/-ABC+DE
```

(ii)

$$A = B + W((I - I_0) + (J - J_0) \times R)$$
$$B = 1025$$
$$W = 4$$
$$I, J = 4, 8$$
$$I_0, J_0 = -6, 4$$
$$R = 10 + 6 + 1 = 17$$
$$A = 1025 + 4 \times ((4 + 6) + (8 - 4) \times 17)$$
$$= 1025 + 4 \times (10 + 68)$$
$$= 1025 + 312$$
$$= 1337$$

(iii) (a)

X = 25

A = 1

$$C = 25 \rightarrow a = a * 10 = 10 \rightarrow c = \frac{25}{10} = 2$$

$$C = 2 \rightarrow a = a * 10 = 100 \rightarrow c = \frac{2}{10} = 0$$

$$X * X = 25 * 25 = 625 \rightarrow 625\%100 = 25 \rightarrow \text{true}$$

(b) Checking whether the number x is. Automorphic Number or not *ic*. Checking whether the number x is present in its square or not.

(iv) **(a)** task(30, 45) → task(30, 5) → task(25, 5) → task(20, 5) → task(15, 5) → task(10, 5) → task(5, 5) → 5

(b) Returning the HCF of two numbers, m and n

3. (i) F(A, B, C, D) = $\sum$(2, 3, 6, 7, 8, 10, 12, 14, 15)

(a)

	C'.D'	C'.D	C.D	C.D'
A'.B'			1	1
A'.B			1	1
A.B	1		1	1
A.B'	1			1

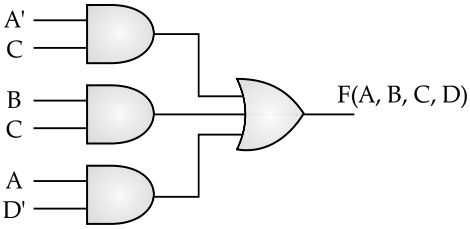
Quad 1(2, 3, 6, 7) = A′.C

Quad 2(6, 7, 14, 15) = B.C

Quad 3(8, 10, 12, 14) = A.D′

F(A, B, C, D) = A′C + BC + AD′

(b)



(ii) F(A, B, C, D) = $\prod$(2, 3, 6, 7, 8, 10, 12, 14, 15)

(a)

	C+D	C+D'	C'+D'	C'+D
A+B	0	0		0
A+B'	0	0		
A'+B'			0	0
A'+B	0		0	0

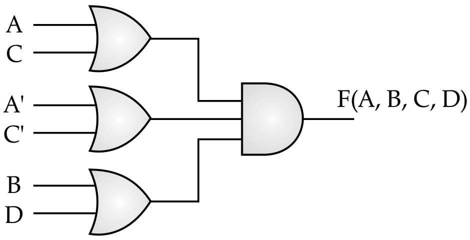
Quad 1(0, 1, 4, 5) = A + C

Quad 2(10, 11, 14, 15) = A′ + C′

Quad 3(0, 2, 8, 10) = B + D

F(A, B, C, D) = (A + C) . (A′ + C′) . (B + D)

(b)



4. (i)

C	A	E	I	X	Maxterm
0	0	0	0	**0**	$C + A + E + I$
0	0	0	1	**0**	$C + A + E + I'$
0	0	1	0	**0**	$C + A + E' + I$
0	0	1	1	**0**	$C + A + E' + I'$
0	1	0	1	**1**	
0	1	0	1	**1**	
0	1	1	0	**0**	$C + A' + E' + I$
0	1	1	1	**0**	$C + A' + E' + I'$
1	0	0	0	**0**	$C + A' + E' + I'$
1	0	0	1	**1**	
1	0	1	0	**1**	
1	0	1	1	**1**	
1	1	0	0	**0**	$C' + A' + E + I$
1	1	1	0	**1**	$C' + A' + E + I'$
1	1	1	0	**1**	
1	1	1	1	**1**	

$X(C, A, E, I) = (C + A + E + I).(C + A + E + I').(C + A + E' + I).(C + A + E' + I').(C + A' + E' + I).$

$(C + A' + E' + I').(C' + A + E + I).(C' + A' + E + I).(C' + A' + E + I')$

$= \prod(0, 1, 2, 3, 6, 7, 8, 12, 13)$

(ii)

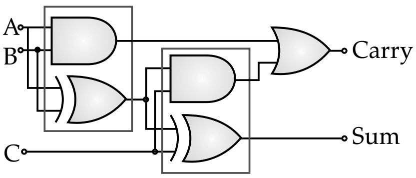
Half Adder	Full Adder
It a adds 2 binary bits	It a adds 3 binary bits
It has 2 inputs	It has 3 inputs
It contains 1 XOR gate and 1 AND gate	It contains 2 XOR gates, 2 AND gates and 1 OR gate

Boolean expression of Full Adder:

$SUM = A \oplus B \oplus C$

$CARRY = (A \oplus B).C + AB$

Logic diagram of Full Adder



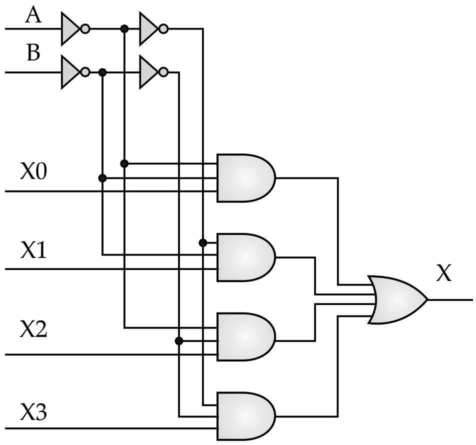
(iii)

A	B	No of l's	$A \odot B$	$(A \odot B)'$	$A \oplus B$
0	0	0	1	0	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	2	1	0	0

Hence, proved $(A \odot B)' = A \oplus B$

5. (i) Encoder is sequential circuit that converts decimal/octal/hexadecimal digit to its equivalent binary form. Decoder is a circuit that does the opposite of Encoder i.e. Binary to the other form.

Logic circuit of 4 : 1 multiplexer



(ii) Gate (1) $= (x.\,y)'$

Gate (2) $= (y + z)'$

$F(x, y, z) = (x.\,y)' + (y + z)'$

(iii) $F(P, Q, R) = (P + Q + R).(P + Q + R').(P + Q' + R').(P' + Q + R)$

6.

```
import java.util.Scanner;
class NumDude
{
    int num;
    public NumDude()
    {
        num=0;
    }
    public void input()
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a no");
        num=sc.nextInt();
    }
    public int sumDigits(int x)
    {
        if(x==0)
        return 0;
        else
        return x%10+sumDigits(x/10);
    }
    public void isDude()
    {
        int n=sumDigits(num);
        if(n*n*n==num)
        System.out.println("Dudeney number");
```

```
        else
        System.out.println("Not a Dudeney number");
    }
    public static void main(String args[])
    {
        NumDude obj=new NumDude();
        obj.input();
        obj.isDude();
    }
}
```

7.
```
import java.util.Scanner;
class Trans
{
    int arr[][];
    int m;
    public Trans(int mm)
    {
        m=mm;
        arr=new int[m][m];
    }
    public void fillarray()
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the array elements");
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
            {
                arr[i][j]=sc.nextInt();
            }
        }
    }
    public void transpose()
    {
        System.out.println("TRANSPOSE");
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
            {
                System.out.print(arr[j][i]+"\t");
            }
            System.out.println();
        }
    }
```

```
        public void display()
        {
            System.out.println("ORIGINAL ARRAY");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print(arr[i][j]+"\t");
                }
                System.out.println();
            }
            transpose();
        }
        public static void main(String args[])
        {
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter the size");
            int mm=sc.nextInt();
            Trans obj=new Trans(mm);
            obj.fillarray();
            obj.display();
        }
    }
```

8.

```
    import java.util.*;
    class SortAlpha
    {
        String sent;
        int n;
        public SortAlpha()
        {
             sent="";
             n=0;
        }
        public void acceptsent()
        {
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter a sentence");
            sent=sc.nextLine().toUpperCase();
        }
        public void sort(SortAlpha P)
        {
            String arr[]=P.sent.split("\\s");
            int len=arr.length;
            for(int i=0;i<len-1;i++)
```

```
        {
            for(int j=0;j<len-1-i;j++)
            {
                if(arr[j].compareTo(arr[j+1])>0)
                {
                    String x=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=x;
                }
            }
        }
        for(int i=0;i<len;i++)
        sent+=arr[i]+" ";
    }
    public void display()
    {
        System.out.println(sent);
    }
    public static void main(String args[])
    {
        SortAlpha obj=new SortAlpha();
        SortAlpha tem=new SortAlpha();
        obj.acceptsent();
        tem.sort(obj);
        obj.display();
        tem.display();
    }
}
```

9. (i)

```
class deQueue
{
    int Qrr[];
    int lim, front, rear;
    ….
    void addFront(int a)
    {
        if(front==0)
        System.out.println("OVERFLOW FROM FRONT");
        else
        Qrr[--front]=a;
    }
    int popFront( )
    {
        if(front==rear)
        return -999;
```

```
        else
        return Qrr[front++];
    }
}
```

(ii) Stack follows LIFO order where as Queue follows FIFO order

10.
```
class Supply extends Demand
{
    int pproduced;
    double prate;
    public Supply(String a, String b, int c, int d, double e)
    {
        super(a,b,c);
        produced=d;
        prate=e;
    }
    public double calculation( )
    {
        return Math.abs(prate*pdemand – prate*pproduce);
    }
    public void display( )
    {
        super.display( );
        System.out.println(calculation( ));
}
}
```

11. (i)
```
void addNode(Node ptr, double ss)
{
    Node s=new Node( );
    s.sal = ss;
    s.next=null;
    for(Node t=ptr; t.next!=null; t=t.next);
    t.next=s;
}
```

(ii) **(a)** A F D G B H E

(b) F and A

(c) Level of E = 3 and level of F =1