ISC BOARD EXAMINATION - 2024
COMPUTER SCIENCE
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 the Principle of duality, the Boolean equation (A+B′) • (A+1)= A + B′ will be equivalent to: **[1]**
- **(a)** (A′+B) • (A′+ 1) = A′ + B
- **(b)** (A • B′)+(A • 0) = A • B′
- **(c)** (A′ • B)+(A′ • 1) = A′ • B
- **(d)** (A′ • B)+(A′ • 0) = A′ • B

(ii) When a sequence of OR, NOT, NOR are connected in series, the logic gate obtained is: **[1]**
- **(a)** AND **(b)** NOT
- **(c)** OR **(d)** XOR

(iii) Idempotence Law states that: **[1]**
- **(a)** X + X = X **(b)** X + X′ = 0
- **(c)** X + X = 1 **(d)** X + X′ = X

(iv) **Assertion:** For proposition ~A =>B its contrapositive is B=> ~ A **[1]**
Reason: Contrapositive is the converse of inverse for any proposition.
- **(a)** Both Assertion and Reason are true, and Reason is the correct explanation for the Assertion.
- **(b)** Both Assertion and Reason are true, but Reason is not the correct explanation for the Assertion.
- **(c)** Assertion is true but Reason is false.
- **(d)** Assertion is false but Reason is true.

(v) The complement of the Boolean expression (P′ · Q) + (R • S′) is: **[1]**
- **(a)** (P′ + Q) • (R′ + S)
- **(b)** (P + Q′) • (R′ + S)
- **(c)** (P′ + Q) • (R + S′)
- **(d)** (P + Q′) • (R + S′)

(vi) Assertion: Recursive data structure follows the **LIFO** principle. **[1]**
Reason: Execution of recursive code follows the concepts of data structure **Queue**.
- **(a)** Both Assertion and Reason are true, and Reason is the correct explanation for the Assertion.
- **(b)** Both Assertion and Reason are true, but Reason is not the correct explanation for the Assertion.
- **(c)** Assertion is true but Reason is false.
- **(d)** Assertion is false but Reason is true.

(vii) State any one use of *interfaces* in Java. **[1]**

(viii) Write the cardinal form of the maxterm X + Y′ +Z

(ix) Write the *canonical* SOP expression for F (A, B) = A <=> B **[1]**

(x) State *any one* difference between **instance variable** and **class variable**. **[1]**

Question 2

(i) Convert the following infix notation to postfix form. **[2]**
(P+Q•R-S)/T•U

(ii) An array ARR [–5.....15, 10....20] stores elements in Row Major Wise with each element requiring 2 bytes of storage. Find the address of ARR [10] [15] when the base address is 2500. **[2]**

(iii) The following function is a part of some class:

```
int jolly(int[ ] x, int n, int m)
{
   if (n < 0)
        return m;
   else if(n<x.length)
          m= (x[n] > m) ?x[n]:m;
   return jolly(x, - -n, m);
}
```

- **(a)** What will be the output of **jolly( )** when the value of **x[ ]= {6, 3, 4, 7, 1}**, **n = 4** and **m = 0**? **[2]**
- **(b)** What function does jolly() perform, apart from recursion? **[1]**

(iv) The following function is a part of some class which is used to find the smallest digit present in a number. There are some places in the code marked by ?1?, ?2?, ?3? which must be replaced by an expression / a statement so that the function works correctly.

```
int small_dig(int n)
{  int min =? 1 ? ;
   while (n ! = 0)
   {
      int q = n / 10
      int r =?2? * 10;
      min = r > min ? 3 ? : r;
      n = q;
   }
   return min;
}
```

(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]**

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) To be recruited as the Principal in a renowned College, a candidate must satisfy any one of the following criteria: **[5]**

- The candidate must be a Postgraduate and should either possess a B.Ed. degree or a teaching experience of more than 15 years.

OR

- The candidate must be an employee of the same college with a teaching experience of more than 15 years.

OR

- The candidate must be a Postgraduate but not an employee of the same college and should have a teaching experience of more than 15 years.

The inputs are:

INPUTS	
P	Candidate is a Postgraduate
S	Candidate is an employee of the same College
E	Candidate has a teaching experience of more than 15 years
B	Candidate possesses a B.Ed. degree

(In all the above cases, 1 indicates yes and 0 indicates no)

Output: **X** Denotes eligibility of a candidate [1 indicates eligibility and - 0 indicates ineligibility in all cases]

Draw the truth table for the inputs and outputs given above and write the SOP expression for **X (P, S, E, B).**

(ii) Reduce the above expression **X (P, S, E, B)** by using 4-variable Karnaugh map, showing the various groups (i.e., octal, quads and pairs).

Draw the logic gate diagram for the reduced expression. Assume that the variables and their complements are available as inputs. **[5]**

Question 4

(i) **(a)** Reduce the Boolean function **F (A,B,C,D) = π (0, 2, 4, 6, 8, 9, 10, 11, 14)** 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) Verify if the following proposition is a Tautology, Contradiction or a Contingency, using a truth table. **[3]**

$((A=>B) \wedge (B=>C))=>(A=>C)$

(iii) Find the complement of the following expression and reduce it by using Boolean laws. **[2]**

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

Question 5

(i) How is a *decoder* different from a *multiplexer*? Draw the logic circuit for 3:8 decoder (Octal decoder). Which multiplexer can be derived from the Octal decoder? **[5]**

(ii) Draw the logic gate diagram for 2-input OR gate using NAND gates only. Show the expression at each step. **[3]**

(iii) Write the canonical form of the cardinal terms, $\mathbf{m_3}$ and $\mathbf{M_5}$ for **F (A, B, C, D).** **[2]**

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]**

Design a class DeciHex to accept a positive integer in decimal number system from the user and display its hexadecimal equivalent.

Example 1: Decimal number = 25 Hexadecimal equivalent = 19

Example 2: Decimal number = 28 Hexadecimal equivalent = 1C 1

Some of the members of the class are given below.

Class name : DeciHex

Data members/instance variables:

num : stores the positive integer

hexa : string to store the hexadecimal equivalent of num

Methods/Member functions:

DeciHex( ) : constructor to initialise the data members with legal initial values

void getNum( ) : to accept a positive integer

void convert(int n) : to find the hexadecimal equivalent of the formal parameter 'n' using the **recursive technique**

void display() : to display the decimal number and its hexadecimal equivalent by invoking the function convert( )

Specify the class **DeciHex** giving details of the **constructor( ), void getNum( ), void convert(int) and void display( ).** Define a **main( )** function to create an object and call all the functions accordingly to enable the task.

Question 7 **[10]**

class **InsSort** contains an array of integers which sorts the elements in a particular order.

Some of the members of the class are given below.

Class name : InsSort

Data members/instance variables:

arr[ ] : stores the array elements

size : stores the number of elements in the array

Methods/Member functions:

InsSort(int s) : constructor to initialise size = s

void getArray() : accepts the array elements

void insertionSort() : sorts the elements of the array in descending order using the **Insertion Sort technique**

double find() : calculates and returns the average of all
indentation the odd numbers in the array
void display() : displays the elements of the array in a sorted order along with the average of all the odd numbers in the array by invoking the function find() with an appropriate message

Specify the class **InsSort** giving details of the **constructor(), void getArray(), Void insertionSort(), double find() and void display()**. Define a **main()** function to create an object and call all the functions accordingly to enable the task.

Question 8

Design a class **Coding** to perform some string related operations on a word containing alphabets only.
Example: Input: "Java" **10**
Output: Original word: Java
J = 74
a=97
v = 118
a= 97
Lowest ASCII code: 74
Highest ASCII code: 118

Some of the members of the class are given below.

Class name : Coding
Data members/instance variables:
Wrd : stores the word
len : stores the length of the word
Methods/Member functions:
Coding() : constructor to initialise the data members with legal initial values
void accept() : to accept a word
void find() : to display all the characters of 'wrd' along with their ASCII codes. Also display the lowest ASCII code and the highest ASCII code, in 'wrd'
void show() : to display the original word and all the characters of 'wrd' along with their ASCII codes. Also display the lowest ASCII code and the highest ASCII code in 'wrd', by invoking the function find( )

Specify the class **Coding** giving details of the constructor(), **void accept(), void find( )** and **void show().** Define a **main( )** function to create an object and call all 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

CardGame is a game of mental skill, built on the simple premise of adding and removing the cards from the top of the card pile.

The details of the class **CardGame** are given below.

Class name : CardGame
Data members/ instance variables:
cards[] : array to store integers as cards
cap : to store the maximum capacity of array
top : to store the index of the topmost element of the array
Methods/Member functions:
CardGame(int cc) : constructor to initialise cap = cc and top=-1
void addCard(int v) : to add the card at the top index if possible, otherwise display the "CARD PILE IS FULL" message
int drawCard() : to remove and return the card from the top index of the card pile, if any, else return the value -9999
void display() : to display all the cards of card pile

(i) Specify the class **CardGame** giving details of the functions **void addCard(int)** and int drawCard(). *Assume that the other functions have been defined.*

The main() function and algorithm need NOT be written. **[4]**

(ii) Name the entity described above and state its principle. **[1]**

Question 10

A super class **EmpSal** has been defined to store the details of an employee. Define a subclass Overtime to compute the total salary of the employee, after adding the overtime amount based on the following criteria.

- If hours are more than 40, then ₹ 5000 are added to salary as an overtime amount.
- If hours are between 30 and 40 (both inclusive), then ₹3000 are added to salary as an overtime amount.
- If hours are less than 30, then the salary remains unchanged

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

Class name **EmpSal**
Data members/instance variables:
empnum : to store the name of the employee
empcode : integer to store the employee code
salary : to store the salary of the employee in decimal
Methods/Member functions:
EmpSal(...) : parameterised constructor to assign values to data members
void show() : to display the details of the employee

Class name **Overtime**
Data members/Instance variables:

hours : integer to store overtime in hours

totsal : to store the total salary in decimal

Methods/Member functions:

Overtime(...) : parameterised constructor to assign values to data members of both the classes

void calSal() : calculates the total salary by adding the overtime indentation amount to salary as per the criteria given above

void show() : to display the employee details along with the total salary (salary + overtime amount)

Assume that the super class ***EmpSal*** *has been defined.*

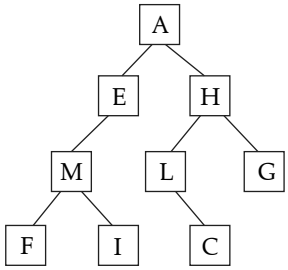
Using the **concept of inheritance**, specify the class **Overtime** giving the details of the **constructor (...), void calSal()** and **void show().**

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

Question 11

(i) With the help of an example, briefly explain the dominant term in complexity. **[2]**

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



(a) Name the external nodes of the tree. **[1]**

(b) State the degree of node M and node L. **[1]**

(c) Write the *post-order* traversal of the above tree structure. **[1]**

■■

ANSWERS

PART I

Answer 1

(i) Option (b) is correct.

Explanation: Principle of Duality states that if we have true Boolean postulates or equations then the dual of this statement equation is also true.

Also to get dual of an expression following changes are done :

+ to .

. to +

1 to 0

0 to 1

(ii) Option (c) is correct.

Explanation: When you connect these gates in series, the output of one gate becomes the input of the next gate.

(A+B)′ = A′B′

(A′B′ +A′B′) ′= (A′B′)′= A+ B

However, you can analyze its behavior based on the operations of each gate:

If the input to the OR gate is true (1), its output will be true (1).

The NOT gate then negates this output, so it becomes false (0).

Finally, the NOR gate receives false (0) as its input. Since both inputs are false (0), the NOR gate outputs true (1).

So, in this specific combination, the output would be true only if the input to the OR gate is true.

(iii) Option (a) is correct.

Explanation: Idempotent Law – An input that is AND´ed or OR´ed with itself is equal to that input · e.g X+X= X , X.X =X

(iv) Option (d) is correct.

Explanation: The contrapositive of a proposition is formed by negating both the consequent (the "then" part) and the antecedent (the "if" part) and then reversing their order. It is logically equivalent to the original proposition.

The contrapositive of the proposition ~A -> B is ~B -> A

(v) Option (b) is correct.

Explanation: (P′Q + RS′)′

(P′Q)′ · (RS′)′

(P″+Q′) · (R′ + S″)

(P+Q′) · (R′+S)

(vi) Option (b) is correct.

Explanation: Going with the implementation of stack , reverse strings etc. a recursive data structure follows the reverse order , last in first out.

(vii) They are used to achieve abstraction, polymorphism, and multiple inheritance.

While getting the Cardinal form , the non complemented variables are written as 0 and the complemented variables are written as 1.

The binary combination formed is then converted to decimal equivalent.

(viii) (X + Y′ + Z): 010

Decimal Equivalent of binary pattern 010: 2

Hence, Cardinal form is M_2

Cardinal form : If an expression is represented in its decimal equivalent of its terms.

(ix) The canonical SOP expression for F(A,B) is: F(A,B)=A′·B′+A·B

Cardinal form: If an expression is represented in its decimal equivalent of its terms.

A⇔B we first need to create a truth table for the function:

A	B	F(A,B)
0	0	1
0	1	0
1	0	0
1	1	1

From the truth table, we can see that F(A, B) is true (1) when either both A and B are false (0) or both A and B are true (1).

Now to construct the SOP expression, we take the minterms (rows in the truth table where the function is true) and negate them. Then, we tave the product (AND) of these terms.

The minterms for which F(A, B) =1 are m0 and m_3:

- m_0: A = 0,B =0, so the term A'·B'.
- m_3: A = 1,B =1, so the term A·B.

(x) The main difference is that instance variables are unique to each instance of a class, while class variables are shared among all instances of the class and are associated with the class itself.

Answer 2

(i) Postfix form: PQR*+S-T/U*

S No.	Expression	Stack	Postfix
0	(	(	
1	P	(	P
2	+	(+	P
3	Q	(+	PQ
4	*	(+*	PQ
5	R	(+*	PQR
6	-	(-	PQR*+
7	S	(-	PQR*+S
8	)		PQR*+S-
9	/	/	PQR*+S-
10	T	/	PQR*+S-T
11	*	*	PQR*+S-T/
12	U	*	PQR*+S-T/U
13			PQR*+S-T/U*

(ii) Number or rows say M = (Ur – Lr) + 1 = [15 – (- 5)] +1 = 21

Number or columns say N = (Uc – Lc) + 1 = [20 – 10)] +1 = 11

Row Major Wise Calculation

The given values are: B = 2500, W = 2 byte, I = 10, J = 15, Lr = -5, Lc = 10, N = 11

Address of A [I][J] = B + W * [N * (I – Lr) + (J – Lc)]

= 2500 + 2* [11 * (10 – (-5)) + (15 – 10)]

= 2500 + 2 * [11 * 15 + 5]

= 2500 + 2 * [165 + 5]

= 2500 + 340

= 2840

(iii) (a) Output: 7

Explanation:

Given the array 'x={ 6, 3, 4, 7, 1}`, `n= 4', and 'm= 0 ', let's compute the output step by step:

1. `jolly(x, 4, 0) `:

- **`n = 4`**, SO **`n <x.length`** is true.
- **`m = (x[4] > m) ? x[4]: m`**, which is **`m (1>0) ? 1 :0`**,so**`m`** becomes **'1'**.
- Recursively call **`jolly(x, 3, 1)'.**

2. `jolly(x, 3, 1)':

- **'n = 3'**, SO **'n <x.length'**is true.
- **'m = (x[3] > m) ? x[3] : m'** , which is **'m = (7>1) ? 7: 1'**, SO **`m`** becomes **'7'**.
- Recursively call **'jolly(x, 2, 7)' .**

3. 'jolly(x, 2, 7)':

- **'n =2`**, so **'n< x. length'** is true.
- **'m= (x[2] > m) ? x [2] : m`**, which is **`m (4>7)?4: 7`**, SO **`m`** remains **'7'.**
- Recursively call **'jolly(x, 1, 7)'.**

4. 'Jolly(x, 1, 7)':

- **'n= 1'**, so **'n<x.length'** is true.
- **'m=(x[1] > m) ? x[1] : m'**, which is **`m = (3>7) ?3 :7'**, so **'m'** remains **'7'.**
- Recursively call **'jolly(x, 0,7)'.**

5. 'jolly(x, 0, 7)':

- **'n = 0,** so **'n < 0'** is false.
- **'n <x.length'** is true.
- **'m= (x[0] > m) ? x[0],** which is **'m (6>7) ?6 :7'**, so **'m'** remains **'7'**.
- Recursively call **'jolly(x, -1, 7)' .**

6. 'jolly(x, -1, 7)':

- **'n =-1**, so **'n <0'** is true, and it returns **'m'**, which is **'7'**.

So, the output of the **'jolly()'** function with the given parameters will be **'7'**.

(b) The jolly() function performs the task of finding the maximum value in a given array x up to index n, while keeping track of the maximum value encountered so far in the parameter m.

(iv) (a) n%10

(b) q

(c) min

SECTION-A

Answer 3

(i)

P	S	E	B	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0

1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

SOP : X(P,S,E,B)= $\sum(6,7,9,10,11,13,14,15)$

$=m_6+ m_1+ m_9+ m_{10}+ m_{11}+ m_{13}+ m_{14}+ m_{15}$

(ii)

PS \ EB	E.'B'	E.'B	EB	EB'
P'S'	0	1	3	2
P'S	4	5	7 1	6 1
PS	12	13 1	15 1	14 1
PS'	1	9 1	11 1	10 1

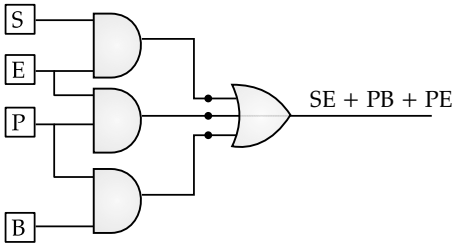
Quad1($m_6 +m_7 +m_{14} +m_{15}$)=SE

Quad2($m_9 +m_{11} +m_{13} +m_{15}$)=PB

Quad3($m_{10} +m_{11} +m_{14} +m_{15}$)= PE

Hence, X(P,S,E,B)=SE + PB + PE

Logic Gate:



Answer 4

(i) (a)

AB \ CD	C+D	C+D'	C'+D'	C'+D
A+B	0 0	1	3	2 0
A+B'	4 0	5	7	6 0
A'+B'	12	13	15	14 0
A'+B	8 0	9 0	11 0	10 0

Logic Gate:

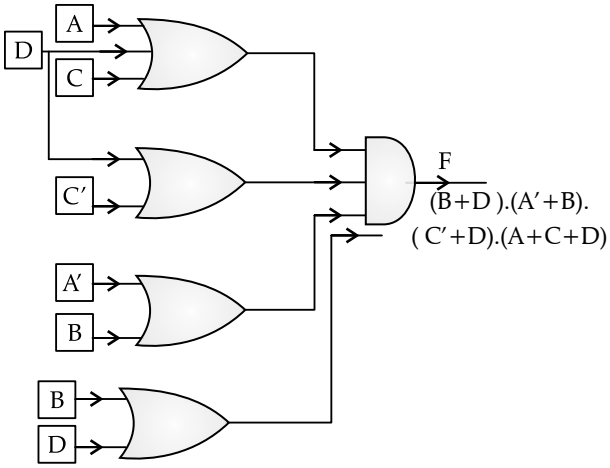
Quad1($M_0 +M_2 +M_8 +M_{10}$)=B+D

Quad2($M_8 +M_9 +M_{11} +M_{10}$)=A'+B

Quad3($M_2 +M_6 +M_{14} +M_{10}$)= C' + D

Pair1($M_0 + M_1$)= A + C + D

Hence, F(A, B, C, D)=(B + D) · (A' + B) · (C'+ D) · (A+C+D)



(ii)

A	B	C	A=>B	B=>C	(A=>B) ^ (B=>C)	A=>C	**((A=>B) ^ (B=>C))=>(A=>C)**
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1

Hence, it is Tautology.

(iii) P·(P+Q) · Q· (Q + R')

Taking complement:

(P.(P+Q).Q.(Q+R'))'

=>P'+(P+Q)' + Q' +(Q+R')'

=>P' + P'Q' + Q' + Q'R

=>P' (1+Q') +Q' (1+R)

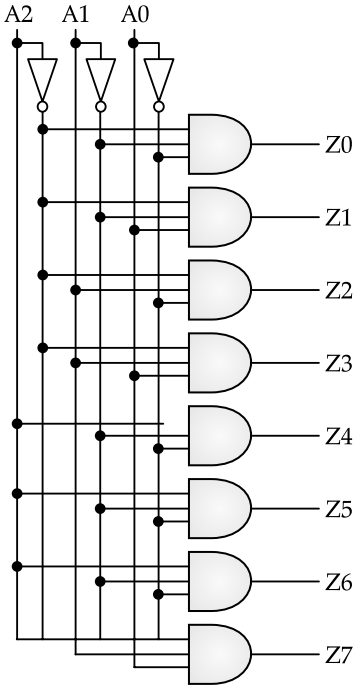
=>P' +Q'

Answer 5

(i) Difference between decoder and multiplexer:

1. Multiplexers transmit data while decoders interpret coded data.
2. Multiplexer is a device which consists of multiple input channels through a single line while decoders consist of multiple inputs passing through multiple output.
3. Multiplexer converts inputs from unary codes (initial) to binary codes while decoder converts binary codes to inputs.
4. There is only 1 output lines in case of multiplexer. A decoder has 2n output lines.

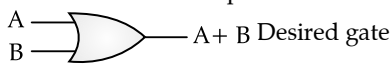
Logic Circuit for 3:8 Decoder:



3 to 8 Decoder Circuit

A multiplexer that can be derived from the Octal decoder is an 8-to-1 multiplexer. This multiplexer has eight input lines (corresponding to the eight output lines of the decoder) and three select lines (corresponding to the three input lines of the decoder). The select lines control which input line is forwarded to the output.

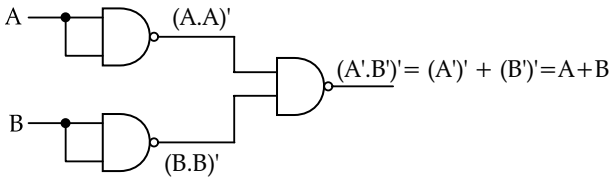
(ii)



A + B = [(A + B)]′ involution

= [(A · B)]′ DeMorgan's

A+ B = [(A · A)′ · (B · B)′]′ Idempotency



[(A · A)′ · (B · B)′]′ = A + B

(iii) Canonical Form:

m_3= 0011 = A′B′CD

M_5 = 0101 = A+B′+C+D′

SECTION - B

Answer 6

```
import java.util.Scanner;
public class DeciHex
{
    int num;
    String hexa;
    char[] hexChar = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A','B',
    'C', 'D', 'E', 'F'};
    String strHex = "";

    // Constructor
        public DeciHex()
        {
            num = 0;
            hexa="";
        }

    // Method to get the decimal number from the user
        public void getNum()
        {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a decimal number:
            ");
            num = scanner.nextInt();
        }

// Method to convert the decimal number to hexadecimal
        public String convert(int n)
        {
            if (n != 0)
            {
                int t = n % 16;
                strHex = hexChar[t] + strHex;
                n = n / 16;
                convert(n);
            }
            return strHex
        }

    // Method to display the decimal number
        public void display()
        {
            System.out.println("Decimal number: " +
            num);
            hexa=convert(num);
            System.out.println("Hexadecimal
            equivalent: " + hexa);
        }

    // Main method
        public static void main(String[] args)
        {
            DeciHex obj = new DeciHex();
            obj.getNum();
            obj.display();
            obj.convert(num);
        }
}
```

Answer 7

```
import java.util.Scanner;
```

```
public class InsSort
{
    double[] arr;
    int size;

// Constructor
    public InsSort(int capacity) {
    array = new double[capacity];
    size = capacity;
}

// Method to get the array elements from the user
    public void getArray()
    {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the elements of the array:");
    for (int i = 0; i < size; i++)
    {
        System.out.print("Element " + (i + 1) + ": ");
        array[i] = scanner.nextDouble();
        size++;
    }
}

// Method to perform insertion sort in descending order
    public void insertionSort()
    {
        for (int i = 1; i <= size; i++)
        {
            double key = array[i];
            int j = i - 1;
            while (j >= 0 && array[j] < key)
            {
                array[j + 1] = array[j];
                j = j - 1;
            }
            array[j + 1] = key;
    }
}

// Method to find the average of the odd array elements
    public double find()
    {
        double sum = 0;
        for (int i = 0; i < size; i++)
        {
            if (array[i]  %2!=0 )
            sum += array[i];
        }
        return (sum / size);
    }

// Method to display the array elements
    public void display()
    {
        System.out.println("Sorted Array in Descending Order:");
        for (int i = 0; i < size; i++)
        {
            System.out.print(array[i] + " ");
        }
        System.out.println();
    }

// Main method
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int size = scanner.nextInt();
        InsSort obj = new InsSort(size);
        obj.getArray();
        obj.insertionSort();
        obj.display();
        double average = obj.find();
        System.out.println("Average of odd array elements: " + average);
    }
}
```

Answer 8

```
    import java.util.Scanner;

    public class Coding
    {
        String wrd;
        int len;

        // Constructor
        public Coding()
        {
            wrd = "";
            len = 0;
        }

    // Method to accept input word from the user
        public void accept()
        {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a word containing alphabets only: ");
            wrd = scanner.next();
            len = wrd.length();
        }
```

```
// Method to display all characters of the word along with their ASCII codes
    public void find()
    {
        int lowest = Integer.MAX_VALUE;
        int highest = Integer.MIN_VALUE;
        System.out.println("Original word: " + wrd);
        for (int i = 0; i < len; i++)
        {
            char c = wrd.charAt(i);
            int ascii = (int) c;
            System.out.println(c + " " + ascii);
            lowest = Math.min(lowest, ascii);
            highest = Math.max(highest, ascii);
        }
        System.out.println("Lowest ASCII code: " + lowest);
        System.out.println("Highest ASCII code: " + highest);
    }

// Method to display the original word and all characters of the word along with their ASCII codes
    public void show( )
    {
    find( ); // Invoking find( ) to display the original word and ASCII codes
    }

// Main method
    public static void main(String[] args)
    {
        Coding obj = new Coding();
        obj.accept();
        obj.show();
    }
}
```

SECTION - C

Answer 9

(i)

```
public class CardGame
{
    int[] cards;
    int cap;
    int top;

// Constructor
    public CardGame(int cc)
    {
        cap = cc;
        cards = new int[cap];
        top = -1;
    }

// Method to add a card to the top of the pile
    public void addCard(int v)
    {
        if (top < cap - 1)
        {
            top++;
            cards[top] = v;
            System.out.println("Card added successfully.");
        }
        else
        {
            System.out.println("CARD PILE IS FULL");
        }
    }

// Method to draw a card from the top of the pile
    public int drawCard()
    {
        if (top >= 0)
        {
            int card = cards[top];
            top--;
            return card;
        }
        else
        {
            System.out.println("No card to draw.");
            return -9999;
        }
    }
//Method to display stack contents
    public void display()
    {
    int t=top;
    if (t >= 0)
    {
        while (t>=0)
        System.out.print(cards[t--] + "→");
    }
    else
    System.out.println("Stack empty");
    }
}
```

(ii) Principle: The principle represented by this entity is the stack data structure principle. In a stack, elements are added and removed from the top only, following the Last In, First Out (LIFO) principle. The addCard() function adds a card to the top of the stack, while the drawCard() function removes and returns the card from the top of the stack. The display() function displays all the cards in the stack, typically from top to bottom.

Answer 10

```
public class Overtime extends EmpSal
{
    int hours;
    double totsal;

// Parameterized constructor
    public Overtime(String empnum, int empcode, double salary, int hours)
    {
        super(empnum, empcode, salary);
        this.hours = hours;
    }

// Method to calculate total salary including overtime
    public void calSal()
    {
        if (hours > 40)
            totsal = salary + 5000;
        else if (hours <=40 && hours >= 30)
            totsal = salary + 3000;
        else
            totsal = salary;
    }
// Method to display employee details along with total salary
    public void show()
    {
        super.show(); // Displaying employee details from superclass
        System.out.println("Total Salary (with overtime): Rs." + totsal);
    }
}
```

Answer 11

(i) The dominant term is the term that grows the fastest. The dominant term in complexity is the term that reflects the main factor determining the algorithm's performance.

For example, n^2 grows faster than n, so if we have something like $g(n) = n^2 + 5n + 6$, it will be big $O(n^2)$. If you have taken some calculus before, this is very similar to the shortcut of finding limits for fractional polynomials, where you only care about the dominant term for numerators and denominators in the end.

(ii) **(a)** External nodes: F, I, C, G

(b) Degree of node M: 2 and L: 1

(c) Post Order Traversal:

F, I, M, E, C, L, G, H, A

■■