# QUALITATIVE ANALYSIS

## Part I (20 Marks)

*Answer **all** questions.*

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

## Question 1

(a) State the *Commutative law* and prove it with the help of a truth table. [1]

(b) Convert the following expression into its canonical POS form: [1]

$$F(X, Y, Z) = (X+Y').(Y'+Z)$$

(c) Find the dual of: [1]

$$(A'+B).(1+B') = A' + B$$

(d) Verify the following proposition with the help of a truth table [1]

$$(P \wedge Q) \vee (P \wedge \sim Q) = P$$

(e) If $F(A, B, C) = A'(BC' + B'C)$, then find $F'$ [1]

### Comments of Examiners

(a) Some candidates only drew the truth table without mentioning the Law. A few candidates were confused with the Associative Law. Many candidates used the 3-variable input truth table instead of the 2-variable input.

(b) Some candidates wrote the direct answer without showing the working. A few candidates got confused with the term 'Canonical' and reduced the expression instead of converting it.

(c) Most of the candidates answered this question correctly. The common mistakes observed were: Compliments were also changed and 0's and 1's were not interchanged. In a few answer scripts, only the definition was written.

(d) Some candidates interchanged the symbols $\wedge$ and $\vee$. Some candidates verified the proposition with Boolean laws instead of Truth Table while a few candidates used the 3-variable truth table instead of the 2-variable truth table.

(e) Some of the candidates wrote the answer directly without showing the working. Change of operators was not done properly by some of the candidates. A few candidates simplified the equation as they were probably not clear with the application of De Morgan's law.

### *Suggestions for teachers*

- Advise students to practice all the laws of Boolean Algebra and Propositional logic.
- Emphasise proving of laws, especially by Truth Table.
- Give adequate practice in the conversion of SOP and POS and vice-versa.
- Explain both Canonical and Cardinal forms laying stress on working/steps.
- Give adequate practice on the principle of duality to show that there exists another equation in every equation.
- Explain, with examples, the difference between compliment and duality.
- Explain the use of the symbols $\wedge$, $\vee$, $\sim$, => and <=> in a proposition.
- Give sufficient practice on complementation using De Morgan's law.

## MARKING SCHEME

## Question 1

(a) Law: **Commutative Law** (i) A + B = B + A (ii) A . B = B. A

Truth Table:

| A | B | A+B | B+A |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

(b) F (X, Y, Z) = (X+Y') . (Y'+Z)
= (X+Y'+0). (Y'+Z+0)
= (X+Y'+ (Z.Z')). (Y'+Z+ (X.X'))
= (X+Y'+Z). (X+Y'+Z'). (X+Y'+Z). (X'+Y'+Z)
= (X+Y'+Z). (X+Y'+Z'). (X'+Y'+Z)

(c) Equation: (A'+ B). (1 +B') = A' + B
**Dual** = A'. B + 0.B' = A'. B

(d) Verify: (P∧Q)∨(P∧~Q) = P

| P | Q | P∧Q | P∧~Q | (P∧Q)∨(P∧~Q) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

(e) F (A, B,C) = A'(BC'+B'C)
F' = A'' + (BC' + B'C)'
= A+ (BC')'. (B'C)'
= A + (B' + C). (B + C')

# Question 2

(a) What are *Wrapper Classes*? Give *any two* examples. [2]

(b) A matrix A[m][m] is stored in the memory with each element requiring 4 bytes of storage. If the base address at A [1][1] is 1500 and address of A [4][5] is 1608, determine the order of matrix when it is stored in **Column Major Wise.** [2]

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

A + (B – C * (D / E) * F)

(d) Define *Big 'O' notation.* State the *two* factors which determine the complexity of an algorithm. [2]

(e) What is *exceptional handling?* Also, state the purpose of *finally* block in a try catch statement. [2]

## Comments of Examiners

(a) The definition written by small number of candidates was vague. Considerable number of candidates wrote only the examples without the definition while a few others wrote the definition without examples. In a few answer scripts the names of wrapper classes were mentioned in the lower case.

(b) Only a few candidates wrote the answer directly without showing the working/formula. Numerous candidates made calculation errors while a few used the Row major instead of the Column major to determine the order of the matrix.

(c) Several candidates wrote the correct answer without showing the working. Some candidates applied the postfix correctly but were unable to derive the final answer due to wrong operator precedence. A few candidates followed order of operation BODMAS while in some answer scripts the concept of left-to-right was not followed.

(d) Some candidates wrote examples to illustrate the answers. Some mentioned the types of complements, while a few other candidates mentioned the cases of complexity (best case, average case and worst case) as examples of factors.

(e) Various answers were given by the candidates for this question. Some candidates wrote syntax to explain *exceptional handling*. A few candidates were unable to understand and state the purpose of *finally* block in a try catch statement and hence, did not attempt writing the answer.

### *Suggestions for teachers*

- Revise the initial topics, with examples, of Class XI syllabus in class XII.
- Elaborate on the basic concepts of Java along with programs.
- Give adequate amount of practice to calculate addresses using Row major and Column major wise.
- Explain the different terms used in address calculations.
- Train students to solve equations.
- Give examples to practice the conversion of Infix to Postfix notation, the order of precedence. Also teach Polish Stack method.
- Train students to practice error handling codes.
- Illustrate to the students, all the three cases of complexities with their factors that influence them.
- Lay emphasis on the difference between factors effecting complexities and types of cases of complexities.

## MARKING SCHEME

## Question 2

| | |
|---|---|
| (a) | **Wrapper Class**: A class which defines a primitive data type. A Wrapper class is a class whose object wraps or contains a primitive data types.<br>**Wrapper class in java** provides the mechanism *to convert primitive into object and object into primitive*.<br>Example: Integer, Character, Double etc. |
| (b) | **Column Major Wise**: $P[i][j] = BA + W [ (j - l_c )* rows + (i - l_r]$<br>Putting the values: $1608 = 1500 + 4[ (4*m + 3)]$<br>$1608 = 1500 + 16m + 12$<br>$16m = 1608-1512$<br>**m = 6** |
| (c) | Infix to Postfix: A + (B – C * (D / E) * F)<br>= A + (B – C * DE/ * F)<br>= A + (B- CDE/* * F)<br>=A + (BCDE/*F*-)<br>Ans: **ABCDE/*F*-+** |
| (d) | **Big 'O' Notation**: The unit of measurement of efficiency of an algorithm. |

| | |
|---|---|
| | A theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n, which is usually the number of items.<br>Two factors are Time and Space(memory) |
| (e) | **Exceptional Handling**: The way to handle anomalous situations during run time.<br>Finally, is a block of code which is executed with or without exceptions. |

# Question 3

The following is a function of some class which checks if a positive integer is a Palindrome number by returning true or false. *(A number is said to be Palindrome if the reverse of the number is equal to the original number.)* The function does not use modulus (%) operator to extract digit. There are some places in the code marked by?**1?**, **?2?**, **?3?**, **?4?, ?5?** which may be replaced by a statement / expression so that the function works properly.

```
boolean PalindromeNum( int N )
 {
        int rev= ?1?;
        int num=N;
        while(num>0)
          {
            int f= num/ 10;
            int s = ?2?;
            int digit = num - ?3?;
            rev= ?4? + digit;
            num /= ?5?;
          }
             if  ( rev= =N )
                 return true;
             else
                 return false;
  }
```

(i) What is the statement or expression at **?1?**
(ii) What is the statement or expression at **?2?**
(iii) What is the statement or expression at **?3?**
(iv) What is the statement or expression at **?4?**
(v) What is the statement or expression at **?5?**

## Comments of Examiners

In sub-parts (i), (ii) and (iii), most of the candidates were able to write the correct answer. Some candidates used the % operator to extract the digit. In sub-part (iv), several candidates wrote only *rev* instead of *rev * 10*. In part (v) a few candidates wrote *num/10* instead of only *10*. Some candidates were not clear in separating digits without the % operator and attempted the code using their own logic and not as per the question.

### *Suggestions for teachers*

- Give adequate practice to the students on program using conditions / looping/recursion and other output related programs.
- Show the dry run/ working of program and instruct the students that working is necessary to get full credit.
- Lay stress on the practice of standard programs.

## MARKING SCHEME

### Question 3

(i) 0
(ii) f*10
(iii) s OR f*10
(iv) rev*10
(v) 10

# 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 4

(a) Given the Boolean function F (A, B, C, D) = Σ (0,2,4,8,9,10,12,13).

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

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

(b) Given the Boolean function: F (A, B, C, D) = $\pi$(3,4,5,6,7,10,11,14,15).

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

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

## Comments of Examiners

(a) (i) Many candidates were unable to draw the correct K-Map for the SOP expression while some candidates drew the POS K-Map instead of SOP. Several candidates used different variables to draw the K-Map instead of the one given in the question. A few candidates included redundant groups in the final expression which was not required.

(ii) Some candidates drew the logic circuit using NAND gates while some others drew vague diagrams with different shapes instead of standard logic gates. Labeling and flow lines were not shown by a few candidates.

(b) (i) Most candidates answered this question correctly. However, the common anomalies found in some answer scripts were:

- Errors in place value and putting variables in K-Map.
- Groups were reduced by laws.
- Incorrect drawing of K-Map.
- Including the redundant group in the final expression.
- Drawing the SOP K-Map instead of POS.

(ii) Some candidates drew the logic circuit using NOR gates while some others drew vague diagrams. Labeling and flow lines were not shown by a few candidates.

### *Suggestions for teachers*

- Teach students to reduce SOP and POS expressions using K-Map simultaneously.
- Give adequate practice to students to draw the K-Map, to fill the K-Map with 0's and 1's, to mark the groups and to reduce the groups.
- Train students to draw logic circuits using basic gates and with universal gates.
- Emphasise on arranging the variables in proper order and the importance of cell values corresponding with the variables.
- Explain clearly how groups are framed and reduced. Tell students that redundant groups are not to be included in the final reduced expression.

## MARKING SCHEME

## Question 4

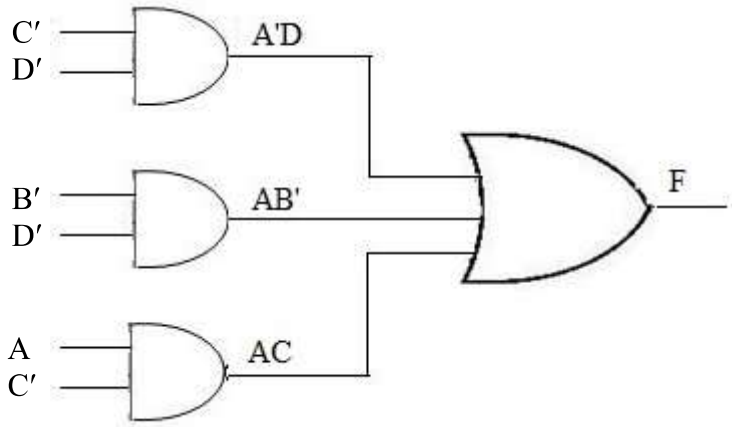
**(a)** $F(A, B, C, D) = \sum (0, 2, 4, 8, 9, 10, 12, 13)$

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

There are three quads:

Quad 1 ($m_0 + m_4 + m_{12} + m_8$) = **C'D'** Quad2 ($m_0 + m_2 + m_8 + m_{10}$) = **B'D'**
Quad 3 ($m_8 + m_9 + m_{12} + m_{13}$) = **AC'**

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



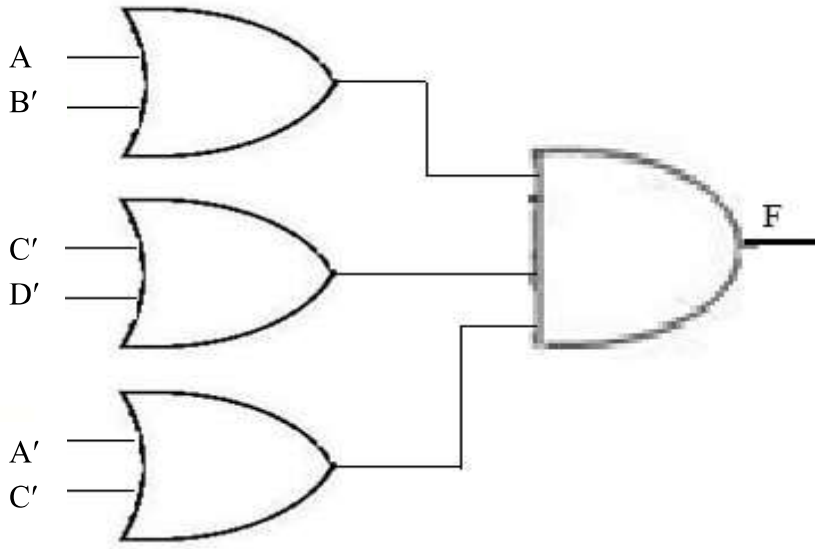

**4(b)** F(A,B,C,D) = **π ( 3 , 4 , 5 , 6 , 7 , 10 , 11 , 14 , 15 )**

| | C+D | C+D' | C'+D' | C'+D |
|---|---|---|---|---|
| A+B | 0 **1** | 1 **1** | 3 0 | 2 **1** |
| A+B' | 4 0 | 5 **0** | 7 0 | 6 **0** |
| A'+B' | 12 1 | 13 1 | 15 0 | 14 0 |
| A'+B | 8 **1** | 9 1 | 11 0 | 10 **0** |

There are three quads:

Quad 1: ($M_4 M_5 M_6 M_7$) = **A + B'** Quad 2: ($M_3 M_7 M_{11} M_{15}$) = **C' + D'**
Quad 3: ($M_{10} M_{11} M_{14} M_{15}$) = **A' + C'**

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




# Question 5

(a) A training institute intends to give scholarships to its students as per the criteria given below: [5]

- The student has excellent academic record but is financially weak.

**OR**

- The student does not have an excellent academic record and belongs to a backward class.

**OR**

- The student does not have an excellent academic record and is physically impaired.

The inputs are:

| **INPUTS** | |
|---|---|
| A | Has excellent academic record |
| F | Financially sound |
| C | Belongs to a backward class |
| I | Is physically impaired |

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

Output: **X** [1 indicates yes, 0 indicates no for all cases]
Draw the truth table for the inputs and outputs given above and write the **SOP** expression for **X (A, F, C, I)**.

(b) Using the truth table, state whether the following proposition is a *tautology*, *contingency* or a *contradiction*: [3]

$$\sim(A \wedge B) \vee (\sim A \Rightarrow B)$$

(c) Simplify the following expression, using Boolean laws: [2]

$$A \cdot (A' + B) \cdot C \cdot (A + B)$$

## Comments of Examiners

(a) Some candidates did not write the final expression. Some candidates were confused with the SOP/POS expression and interchanged 0's and 1's and *vice-versa*. A few candidates reduced the expression using K-Map which was not required. Decoding criteria was not clear to some of the candidates.

(b) Some candidates were confused by the symbols ~, ∧, ∨ and =>. Some candidates used the truth table for
3-variables instead of 2-variables. In a few answer scripts, one column (=>) was incorrect and so the result was also incorrect. Some candidates proved using law instead of truth table while a few did not mention the proposition being a tautology.

(c) Some candidates either gave only the final answer without showing the working/steps or mentioning the laws or lacked clarity in the order of precedence of operators.

### *Suggestions for teachers*

- Give adequate practice to the students in truth table with 4 input variables (i.e. 16 combinations).
- Explain propositional logic to find the criteria for the output.
- Ask students to write the final expression in either Canonical or Cardinal form for both SOP and POS expressions.
- Advise students to practice all the laws of Boolean algebra and Propositional logic.
- Emphasize proving of all the laws.
- Explain the use of the symbols Λ, V, ~, => and <=> in a proposition. Make students practice the concept of tautology, contingency or a contradiction and truth tables for conditional (=>) and bi-conditional (<=>).
- Ensure sufficient practice to students in reducing / simplifying expressions involving maximum laws.
- Encourage students to mention laws along with their working.

## MARKING SCHEME

## Question 5

(a)

| A | F | C | I | X (OUTPUT) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | **0** |
| 0 | 0 | 0 | 1 | **1** |
| 0 | 0 | 1 | 0 | **1** |
| 0 | 0 | 1 | 1 | **1** |
| 0 | 1 | 0 | 0 | **0** |
| 0 | 1 | 0 | 1 | **1** |
| 0 | 1 | 1 | 0 | **1** |
| 0 | 1 | 1 | 1 | **1** |
| 1 | 0 | 0 | 0 | **1** |
| 1 | 0 | 0 | 1 | **1** |
| 1 | 0 | 1 | 0 | **1** |
| 1 | 0 | | | 1 | 1 | **1** |
| 1 | 1 | | | 0 | 0 | **0** |
| 1 | 1 | | | 0 | 1 | **0** |
| 1 | 1 | | | 1 | 0 | **0** |
| 1 | 1 | | | 1 | 1 | **0** |

**SOP Expression: X (A ,F, C, I) = Σ (1, 2, 3, 5, 6, 7, 8, 9, 10, 11)**

**OR** **X = A'F'C'I + A'F'CI' + A'F'CI + A'FC'I + A'FCI' + A'FCI + AF'C'I' + AF'C'I + AF'CI' + AF'CI**

(b) **~(A∧B) ∨ (~A=>B) is a Tautology, Contradiction or Contingency**

| A | B | ~A | ~B | ~(A∧B) | ~A=>B | ~(A∧B) ∨ (~A=>B) |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | **1** | **0** | **1** |
| 0 | 1 | 1 | 0 | **1** | **1** | **1** |
| 1 | 0 | 0 | 1 | **1** | **1** | **1** |
| 1 | 1 | 0 | 0 | **0** | **1** | **1** |

**Hence, it is a Tautology**

(c) Simplify : A.(A'+B).C.(A+B)
= (AA' + AB).(AC + BC)
= (0 + AB).(AC + BC)
= ABC + ABC
= ABC

# Question 6

(a) What is an *Encoder?* Draw the Encoder circuit to convert A-F hexadecimal numbers to binary. State an application of a Multiplexer. [5]

(b) Differentiate between *Half Adder* and *Full Adder*. Draw the logic circuit diagram for a Full Adder. [3]

(c) Using only NAND gates, draw the logic circuit diagram for A′ + B. [2]

## Comments of Examiners

(a) Some candidates defined *decoder* instead of *encoder*. A few candidates drew the entire hexadecimal encoder instead of only A – F. Some candidates, neither named nor labeled the input lines. Several candidates drew the multiplexer diagram to illustrate examples. In a few answer scripts, multiplexer was defined as 'conversion from one form to another'.

(b) In some answer scripts, either the differences were vague or the term 'adds' was missing in the difference. Some candidates drew the truth table and expression for full adder which was not required. A few candidates drew two different diagrams for sum and carry. Some candidates, instead of drawing the logic circuit, drew the block diagram.

(c) Some candidates converted the expression and then drew the circuit using NAND gates. Some

### *Suggestions for teachers*

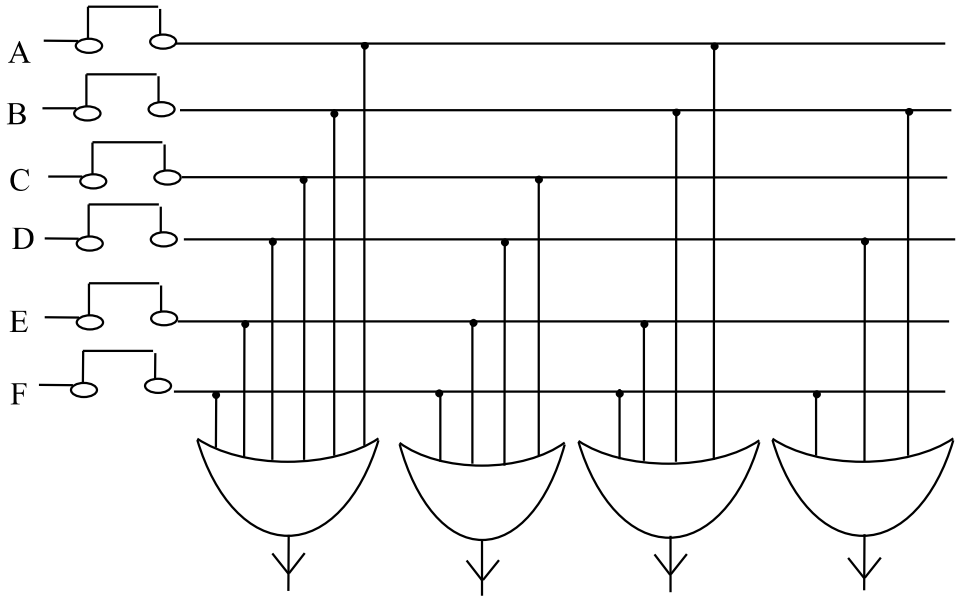
- Give sufficient practice to the students so that they are conversant with the circuit diagram, truth table, expression, definition, and use of all applications of Boolean algebra i.e. Half adder, Full adder. Encoders, Decoders etc.
- Explain the purpose and working of the gates for the said applications.
- Train students to get basic gates using only Universal gates (NAND and NOR).
- Ensure in-depth knowledge of Universal gates.

candidates took the inputs directly as A and B. A few candidates simply replaced the basic gates with NAND gates.

## MARKING SCHEME

## Question 6

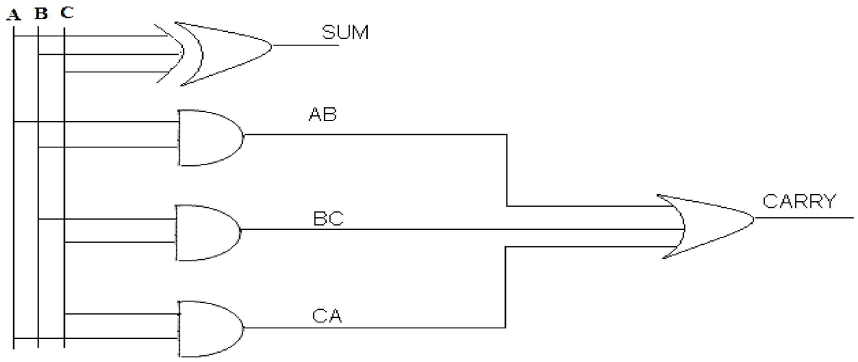
(a) An **Encoder** is a combinational circuit which inputs $2^n$ or fewer lines and outputs n lines.
Circuit diagram of a A – F Encoder:



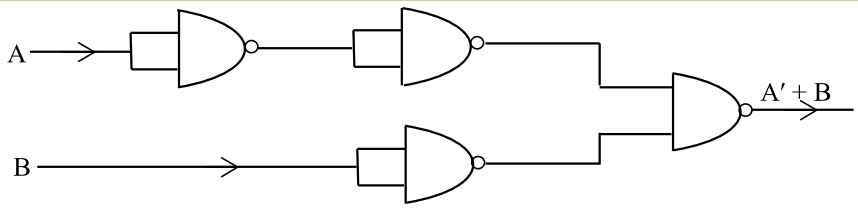

**Application of a Multiplexer:** 1. Used for Routing of signals
2. Data Transmission
3. Telephone Exchanges / TV etc

(b) **Half Adders** : It is a combinational circuit which adds two input binary bits and outputs two binary bits.
**Full Adders** : It is a combinational circuit which adds three input binary bits and outputs two binary bits.

Circuit of a Full Adder : SUM=A⊕B⊕C
CARRY = AB + BC + AC




(c) Using NAND - circuit of A′ + B




# SECTION – B

*Answer **any two** questions.*

## Question 7

Design a class **Perfect** to check if a given number is a perfect number or not. [ A number is said to be perfect if sum of the factors of the number excluding itself is equal to the original number] [10]

Example : 6 = 1 + 2 + 3 (where 1, 2 and 3 are factors of 6, excluding itself)

Some of the members of the class are given below:

| | | |
|---|---|---|
| **Class name** | : | **Perfect** |
| **Data members/instance variables:** | | |
| num | : | to store the number |
| **Methods/Member functions:** | | |
| Perfect (int nn) | : | parameterized constructor to initialize the data member num=nn |
| int sum_of_factors(int i) | : | returns the sum of the factors of the number(num), excluding itself, using **recursive technique** |
| void check() | : | checks whether the given number is perfect by invoking the function *sum_of_factors()* and displays the result with an appropriate message |

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

### Comments of Examiners

Some candidates were not clear on the concept of recursion and the parameters in the method sum of factors (). A few candidates used the additional instance variables which was not in accordance with the question. In some answer scripts, the recursive case was improper as several candidates attempted the recursive method without using the recursive technique and instead the used iteration (loops).

Some candidates missed the base case. Some had problem in check() method as they did not invoke the sum of factors() method properly, although the other methods including the constructor were well answered. Some candidates either did not write the

### *Suggestions for teachers*

- Give lots of practice to the students to solve programs using recursive techniques.
- Explain thoroughly recursion and its techniques with examples.
- Ensure that the students are well conversant with the base case and recursive case for every program using recursive technique.

main () method or were unable to properly do the object creation and method calling in the main() function. Documentation/comments were missing in a few answer scripts.

- Train students Invoking function within another function.
- Explain the difference between iteration and recursion.
- Ensure that the students have understood instance variables and their accessibility
- Advise students to read the question carefully and answer according to its requirement.

## MARKING SCHEME

## Question 7

```
import java.util.*;
class Perfect
{
   int num;
   Perfect(int nn)
   {
      num=nn;
    }
   int sum_of_factors(int i)
    {
      if (i == num)
           return 0;
      else if (num%i==0)
           return i + sum_of_factors(i+1);
       else
            return sum_of_factors(i+1);
     }
     void check()
      {
            int s=sum_of_factors(1);
            if (s==num)
                System.out.println(num+"is a perfect number");
            else
                System.out.println(num+"is not a perfect number");
          }
      public static void main()
      {
         Scanner sc=new Scanner(System.in);
         System.out.println("Enter a number");
         int no=sc.nextInt();
         Perfect ob=new Perfect(no);          // OR  Perfect ob=new Perfect(6);
         ob.check();                          //          ob.check();
      }
   }
```

# Question 8

Two matrices are said to be equal if they have the same dimension and their corresponding elements are equal. **[10]**
For example, the two matrices A and B given below are equal:

Matrix A

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 4 | 5 |
| 3 | 5 | 6 |

Matrix B

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 4 | 5 |
| 3 | 5 | 6 |

Design a class **EqMat** to check if two matrices are equal or not. Assume that the two matrices have the same dimension.

Some of the members of the class are given below:

| | | |
|---|---|---|
| **Class name** | : | **EqMat** |
| **Data members/instance variables:** | | |
| a[ ][ ] | : | to store integer elements |
| m | : | to store the number of rows |
| n | : | to store the number of columns |
| **Member functions/methods:** | | |
| EqMat(int mm, int nn) | : | parameterised constructor to initialise the data members m = mm and n = nn |
| void readarray( ) | : | to enter elements in the array |
| int check(EqMat P, EqMat Q) | : | checks if the parameterized objects P and Q are equal and returns 1 if true, otherwise returns 0 |
| void print( ) | : | displays the array elements |

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

## Comments of Examiners

Most candidates defined *Constructor* properly. However, some candidates did not declare memory to the array using the *new* operator and a few others passed two different arrays instead of objects. In a few answer scripts, the passing of objects was not done correctly. Several candidates checked the size of the matrix which was not required. Some candidates either did not properly do the object creation in the main () method or created three objects instead of two in the main (). Even calling the method check () was incomplete in some answer scripts. Some candidates found the returning type of the method check () a bit confusing and returned true/false instead of 0 or 1.

### *Suggestions for teachers*

- Give a lot of practice in passing of objects to a function through parameters
- Explain the working on one dimensional and two-dimensional arrays with various examples.
- Illustrate pass by value and pass by reference and make the students practice it.
- Teach dynamic binding concepts and dot (.) operator
- Explain the use of flag variables.

## MARKING SCHEME

## Question 8

```
import java.util.*;
class EqMat
{
   int a[][];
   int m;
   int n;
   static Scanner sc=new Scanner(System.in);
   EqMat(int mm,int nn)
   {  m=mm;
      n=nn;
      a=new int[m][n];
   }
   void readarray()
   {  System.out.println("enter" + (m*n) + "elements" );
      for (int i=0;i<m;i++)
         for (int j=0;j<n;j++)
            a[i][j]=sc.nextInt();
     }
    int check(EqMat P,EqMat Q)
    {   for (int i=0;i<P.m;i++)
          for (int j=0;j<P.n;j++)
            {   if (P.a[i][j]!=Q.a[i][j])
                   return 0;
            }
         return 1;
      }
      void print()
      {  for (int i=0;i<m;i++)
          { System.out.println();
            for (int j=0;j<n;j++)
               System.out.print(a[i][j]+"\t");
          }
        }
      public static void main()
      {  EqMat ob1=new EqMat(3,3);
         EqMat ob2=new EqMat(3,3);
         System.out.println("enter nos for the Ist Matrix");
         ob1.readarray();
         System.out.println("enter nos for the 2nd Matrix");
         ob2.readarray();
         if (ob1.check(ob1,ob2)==1)
          { System.out.println("Equal Matrix");
            ob1.print();
            ob2.print();
          }
         else
            System.out.println("not equal");
        }
      }
```

# Question 9

A class **Capital** has been defined to check whether a sentence has words beginning with a capital letter or not. **[10]**

Some of the members of the class are given below:

| | | |
|---|---|---|
| **Class name** | : | **Capital** |
| **Data member/instance variable:** | | |
| sent | : | to store a sentence |
| freq | : | stores the frequency of words beginning with a capital letter |
| **Member functions/methods:** | | |
| Capital( ) | : | default constructor |
| void input() | : | to accept the sentence |
| boolean isCap(String w) | : | checks and returns true if word begins with a capital letter, otherwise returns false |
| void display () | : | displays the sentence along with the frequency of the words beginning with a capital letter |

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

## Comments of Examiners

Some candidates used different methods/logic to split the sentence into words. Several candidates used blank space, while some others used the split () method to separate the words in the sentence. Different logics were used to check the upper case. In a few answer scripts, the ASCII values were incorrect. Some candidates did the splitting in the main () method while some others did it in isCap () method itself.

### *Suggestions for teachers*

- Give practice to the students to extract characters from words, words from sentences and sentences from paragraphs.
- Adopt different methods /logic so that the students get a wider exposure to string manipulation related programs.
- Ensure that the students have the knowledge of constructors to initialize a string and other data members and have been given adequate practice in this topic.
- Give adequate practice in conversion of string into characters and concatenating of strings.

## MARKING SCHEME

## Question 9

```
import java.util.*;
class Capital
{
   String sent;
   int freq;
   Capital()
   {
      sent="";
      freq=0;
   }
   void input()
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter the sentence");
      sent=sc.nextLine();
   }
   boolean isCap(String w)
    {
      char c=w.charAt(0);
      if (Character.isUpperCase(c) )
           return true;
       else
            return false;
      }
    void display()
     {   System.out.println("sentence="+sent);
         StringTokenizer ob=new StringTokenizer(sent);
         int count=ob.countTokens();
         for (int i=0;i<count;i++)
            {  String wd=ob.nextToken();
               if (isCap(wd))
                  freq=freq+1;
             }
            System.out.println("Freqency of title word="+freq);
     }
```

**OR**

```
    void display()
          {   System.out.println("sentence="+sent);
              String b[]=sent.split(" ");
              for(int i=0;i<b.length;i++)
               {  if(isCap(b[i]))
                     freq++;
               }
                System.out.println("Freqency of title word="+freq);
           }
```

**OR**

```
    void display()
```

```
    {   System.out.println("sentence="+sent);
        String b=" "+sent;
        char c;
        for(int i=0;i<b.length();i++)
         {    c=b.charAt(i);
              if (c==' ')
               { if(Character.isUpperCase(b.charAt(i+1)) )
                     freq++;
               }
          }
           System.out.println("Freqency of title word="+freq);
        }

    public static void main()
    {
       Capital ob=new Capital();
       ob.input();
       ob.display();
     }
}
```

# SECTION – C

*Answer **any two** questions.*

## Question 10

A super class **Number** is defined to calculate the factorial of a number. Define a sub class **Series** to find the sum of the series S = 1! + 2! + 3! + 4!+…….....+n! **[5]**

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

| | | |
|---|---|---|
| **Class name** | : | **Number** |
| **Data member/instance variable:** | | |
| n | : | to store an integer number |
| **Member functions/methods:** | | |
| Number(int nn) | : | parameterized constructor to initialize the data member n=nn |
| int factorial (int a) | : | returns the factorial of a number (factorial of n = 1×2×3×……×n) |
| void display() | : | displays the data members |
| **Class name:** | | **Series** |
| **Data member/instance variable:** | | |
| sum | : | to store the sum of the series |
| **Member functions/methods:** | | |
| Series(…) | : | parameterized constructor to initialize the data members of both the classes |
| void calsum( ) | : | calculates the sum of the given series |

void display( ) : displays the data members of both the classes

*Assume that the super class* ***Number*** *has been defined*. Using the **concept of inheritance**, specify the class **Series** giving the details of the **constructor(…),void calsum( )** and **void display( )**.

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

## Comments of Examiners

Most of the candidates wrote the program correctly. However, the common anomalies found in some answer scripts were:

- Vague concept of Inheritance.
- Missing keywords *extends* and *super*.
- Constructor with inheritance was not answered correctly.
- the data members were not declared properly.
- The function calsum( ) in the derived class was not answered properly.
- No loop being used to calculate the sum of the series.
- Algorithm being written instead of a program. Some candidates were not clear about accessing the members of the super class by the derived class. A few candidates declared the base class also, which was not required.

## *Suggestions for teachers*

- Explain the importance of the keywords *extends* and *super* in inheritance and Concept of overriding in inheritance with examples.
- Give adequate practice to the students on the concept of Inheritance.
- Illustrate the use of constructor of the base class. Explain to the students the different visibility modes and their accessing capability.
- Ensure that the students have the knowledge of calling the member function from the super class to the derived class.
- Instruct students to read the question attentively and answer accordingly. must be explained with examples.

## MARKING SCHEME

## Question 10

```
class Series extends Number
{
   long sum;

   Series(int nn)
   {
      super(nn);
      sum=0;
     }

     void calsum()                                  void calsum()
     {  for (int i=1;i<=n;i++)          OR          {  int f=1;
            sum=sum+factorial(i);                     for (int i=1;i<=n;i++)
```

```
    }                                                    {  f=f*i ;
                                                           sum=sum + f ;
                                                        }
                                                     }
     void display()
     {
        super.display();
        System.out.println("sum of the series="+sum);
     }
}
```

# Question 11

**Register** is an entity which can hold a maximum of 100 names. The register enables the user to add and remove names from the top most end only.

Define a class Register with the following details:

**Class name** : Register

**Data members / instance variables:**

| | |
|---|---|
| stud[ ] | array to store the names of the students |
| cap | stores the maximum capacity of the array |
| top | to point the index of the top end |

**Member functions:**

| | |
|---|---|
| Register (int max) | constructor to initialize the data member cap = max, top = −1 and create the string array |
| void push(String n) | to add names in the register at the top location if possible, otherwise display the message "OVERFLOW" |
| String pop() | removes and returns the names from the top most location of the register if any, else returns "$$" |
| void display() | displays all the names in the register |

(a) Specify the class **Register** giving details of the functions **void push(String)** and **String pop( ).** Assume that the other functions have been defined. **[4]**

**The main function and algorithm need NOT be written**.

(b) Name the entity used in the above data structure arrangement. **[1]**

## Comments of Examiners

(a) The concept of stack was not clear to most of the candidates. The common errors made by some of the candidates were:

- the condition / logic for underflow and overflow was not answered correctly.
- increment / decrement of top index was not done properly.
- used the stack to store integers instead of string.
- the methods push() and pop() were found to be difficult.

A few candidates used all the methods which were not required.

(b) Some candidates were unable to name the entity and wrote vague answers. A few candidates over looked this part while some mentioned the principles LIFO and FIFO as entities.

### *Suggestions for teachers*

- Give more practice to the students in data structure programs like the stacks, queues, dequeue etc.
- Do the working as to how the stack or a queue performs and support the workings with examples.
- Explain the concept of LIFO and FIFO with real life examples.
- Explain the concept taking the base as an array.
- Emphasise the implementation of stacks, queues and de queues using arrays.
- Illustrate that an array related program cannot be manipulated by shifting / inserting or initializing by any value since these data structures require pointers.
- Clarify students that pointers are not supported in java. Hence, the array is used to show the working of a stack, queue or a dequeue.

## MARKING SCHEME

## Question 11

(a)
```
class Register
    {
        void push(String n)
            {    if (top<cap-1)
                 stud[++top]=n;
                 else
                 System.out.println("OVERFLOW");
            }
         String pop()
          { if (top>=0)
              return stud[top--];
         else
              return " $$";
          }
       }
```

(b) The entity used in the above data structure is **Stack**.

# Question 12

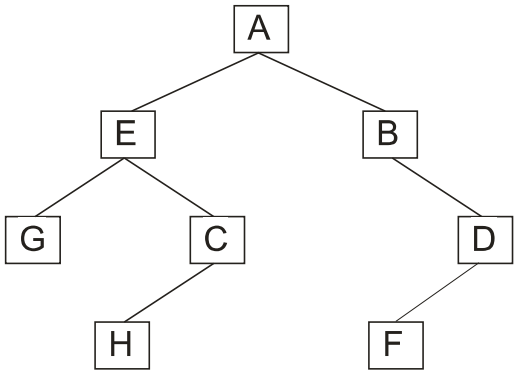
(a) A linked list is formed from the objects of the class **Node**. The class structure of the Node is given below: [2]

```
class   Node
{
        int n;
        Node   link;
}
```

Write an *Algorithm* **OR** a *Method* to search for a number from an existing linked list.
The method declaration is as follows:

```
void  FindNode( Node  str, int b  )
```

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




(i) Write the inorder traversal of the above tree structure. **[1]**

(ii) State the height of the tree, if the root is at level **0** (zero). **[1]**

(iii) List the leaf nodes of the tree. **[1]**

## Comments of Examiners

(a) Many candidates answered this part correctly. However, some candidates had problem in moving the pointer to the next node and checking for null. Several candidates did not create the temporary pointer. Loop was also missing in a few answer scripts. Some candidates wrote the algorithm in simple English language covering all the main steps.

(b) (i) Some candidates wrote the other orders instead of inorder traversal of the given tree. In a few answer scripts, one or two nodes were not placed correctly.

(ii) Some candidates wrote the height of the tree as 4 instead of 3.

(iii) Most of the candidates answered this part correctly barring a few exceptions who could not list the leaf nodes accurately.

### *Suggestions for teachers*

- Allow students to practice more methods / algorithms with link list and binary tree data structure.
- Train students to use the diagrams to illustrate the link list and the Binary Trees.
- Ask students to revise frequently the concepts related to temporary pointer, checking for null condition and moving pointer to the next node etc.
- Illustrate the concept of root, height, depth, size, degree, siblings, nodes (internal and external), levels, tree traversals etc. thoroughly by using a binary tree diagram.

## MARKING SCHEME

## Question 12

(a) ALGORITHM:

Step 1. Start
Step 2. Set temporary pointer to the first node
Step 3. Repeat steps 4 and 5 until the pointer reaches null. Display number not found
Step 4. check for number , if found display, exit
Step 5. Move pointer to the next node
Step 6. End algorithm

METHOD:

```
void FindNode(Node str, int b)
  {
     Node temp=str;
     while(temp!=null)
       { if (temp.n == b)
           {System.out.prinln(b+" is found ");
            break;
           }
         temp=temp.link;
       }
  }
```

(b) (i) G,E,H,C,A,B,F,D

(ii) 3

(iii) G,H,F

# GENERAL COMMENT

## Topics found difficult by candidates

- Wrapper classes
- Extracting digits from a number without using the % operator.
- K-MAPS (grouping, redundant groups, map-rolling, place value).
- Recursive technique:
  - Passing objects to functions.
  - Stack operations for push() and pop ( ) methods.

## Concepts in which candidates got confused

- The symbols ('=>', 'ʌ' and 'v' )in a proposition.
- Complexity and factors.
- *Exceptional handling* and *finally* block in a try catch statement.
- Output without using the % operator.
- Passing objects to functions.
- Splitting words from a sentence.
- Passing of objects.
- Use of single instance variable for multiple operations in various functions.
- Link list and Stacks.
- Height of tree.

## Suggestions for candidates

- Refer to the reliable resources [text books, surf the net (Wikipedia etc.)] for latest concepts and improved techniques of programming.
- Download directly proper definitions, output programs, algorithms etc. from the net.
- Prepare summary for each chapter.
- Underline/highlight important words, terms, definitions etc.
- Practise on the system on a regular basis to understand the syntax and to correct errors.
- Mention the Laws while reducing a Boolean Expression.
- Practise one form of K-Map with proper place value for both SOP and POS.
- Documentation is compulsory in programming and should be mentioned with each program.
- Declare the class with data members and member functions.
- Expand or define each function according to the instructions given by the side of each function.
- Understand the logic of writing a program.
- Practise constructors with every program.
- Treat each function of a class as a separate program.
- Scope of the syllabus should be followed.
- Prepare section wise following the scope of the syllabus.
- Solve previous years' ISC question papers.
- Workings, wherever required, should be shown on the side of each question.
- Always answer to any question as per the requirement of the question.