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 law represented by the following proposition and prove it with the help of a truth table: **[1]**

P V P = P

(b) State the Principle of Duality. **[1]**

(c) Find the complement of the following Boolean expression using De Morgan's law: **[1]**

F(a,b,c) = (b′ + c) + a

(d) Draw the logic diagram and truth table for a 2 input XNOR gate. **[1]**

(e) If (~P => Q) then write its: **[1]**

(i) Inverse

(ii) Converse

Comments of Examiners

(a) Most of the candidates answered this part well. Some mentioned the laws involving addition while some did not. Some confused the symbol V with the symbol Λ. A few candidates used the truth table using 3 variables instead of 2 variables. Others proved by Boolean law instead of the truth table.

(b) Some candidates gave an example to illustrate the Principle of duality. Others did not mention that the complements remain unchanged.

(c) Several candidates wrote the answer directly without showing the working. Change of operators was not properly done by some of the candidates.

(d) A number of candidates drew the circuit instead of the gate symbol. In some cases, XOR gate was drawn instead of XNOR.

(e) Some candidates were confused with the symbols '= >' and ' ~ ' while others interchanged the answers. A few candidates proved it with the help of Boolean laws.

Suggestions for teachers

- Candidates should be told to practice all the laws of Boolean algebra and Propositional logic. Proving of all the laws must be emphasized. The use of the symbols Λ, V, ~, => and <=> in a proposition must be explained.
- Difference between complement and duality must be explained with examples.
- More practice on complementation using De Morgan's law should be given for such type of questions.
- All the gates of Boolean algebra must be practiced with their respective gate symbols, truth table, use, performance and expression.
- Proportional logic should be taught using all terms that are required. The symbols used in proportions must be explained.

MARKING SCHEME

Question 1

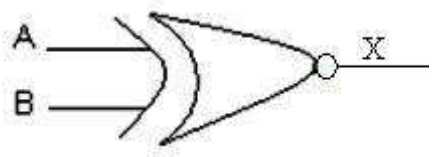
(a) Law : **Idempotent Law**. It states that $P \vee P = P$ and $P \wedge P = P$.
Truth Table:

P	P	P ∨ P
0	0	0
1	1	1

(b) **Principle of Duality** states that, to every Boolean Equation there exists another equation which is dual to the original equation. To achieve this, the AND's (.) are converted to OR's (+) and vice-versa, 0's to 1's and vice versa, however, the complements remain unchanged.

(c) Complement of : F(a,b,c) = (b' + c) +a
= (b' + c)' . a'
= b'' . c' .a' = b . c' . a'

(d) XNOR gate :



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

(e) If (~ P => Q) then ;

(i) **Inverse** : (P => ~ Q) OR P' + Q'
(ii) **Converse** : (Q => ~ P) OR Q'+ P'

Question 2

(a) What is an *interface*? How is it different from a *class*? **[2]**

(b) Convert the following infix expression to postfix form: **[2]**

P * Q / R + (S + T)

(c) A matrix P[15][10] is stored with each element requiring 8 bytes of storage. If the base address at P[0][0] is 1400, determine the address at P[10][7] when the matrix is stored in Row Major Wise. **[2]**

(d) (i) What is the worst case complexity of the following code segment: **[2]**

```
for (int x = 1; x <=a; x++)
  {
      statements;
  }
for (int y = 1; y <=b; y++)
  {
    for (int z = 1; z <=c; z++)
        {
          statements;
        }
  }
```

(ii) How would the complexity change if all the three loops went to N instead of a, b and c?

(e) Differentiate between a *constructor* and a *method* of a class. **[2]**

Comments of Examiners

(a) This part was answered well by most of the candidates. Some wrote vague definitions of *interface*. Others explained using examples. Some candidates used the keywords 'extends' and 'implements' to differentiate an *interface* from a *class*.

(b) Most candidates were able to solve this problem correctly. Several candidates wrote the correct answer without showing the working. Some applied the postfix correctly, but could not derive the final answer due to wrong operator precedence. BODMAS was followed in some cases instead of left-to-right.

(c) Some candidates wrote the answer directly without showing the working/formula. Calculation mistakes were also observed. Others did by Column major instead of Row major.

(d) (i) This part was well answered by almost all candidates. Only a few did not mention O(a) in the final complexity O(a + bc). Dominant term was not clear in some cases.

(ii) A few candidates were not able to answer the change in complexity. The dominant term was not clear in some cases. A few candidates wrote N^3 instead of N^2 in the final answer.

(e) Various answers were given by candidates. Some explained with the help of examples. Others wrote the definition of both, without mentioning the differences.

Suggestions for teachers

- Inheritance with interface and classes must be given more practice. The concept of multiple inheritance must be explained using an interface.
- Examples need to be practiced with conversion of Infix to Postfix notation, the order of precedence. The Polish Stack method must also be taught.
- More practice should be given to calculate addresses using Row major and Column major wise. The different terms used in address calculations must be explained.
- Complexity and Big 'O' notation must be given more practice. Examples using loops, nested loops and conditional statements must be solved and explained.
- The difference between the two terms 'constructor' and 'method' must be clarified. This will enable students to understand the concepts and their differences clearly.

MARKING SCHEME

Question 2

(a) **Interface** is a non primitive data type which has static and final data members and prototype of functions (i.e. functions are not defined)
Difference : Interface supports multiple inheritance whereas a Class does not support multiple Inheritance

(b) Infix to postfix : P * Q / R + (S + T)
= P * Q / R + ST+
= P Q* / R + ST+
= PQ*R/ + ST+
Ans : **P Q * R / S T + +**

(c) **Row Major Wise**: P[i][j] = BA + W [(i – l_r)* column + (j – l_c]
= 1400 + 8[(10-0)*10 + (7-0)]
= 1400 + 856
P[10][7] = 2256

(d) (i) O(a) + O(b x c)
= O(a + bc)
(ii) O(N) + O(N^2)
= O(N^2) taking the dominant term.

(e) Constructor has the same name of the class where as a method has a different name. There is no returning type, not even void in constructor where as in a method it can return a value.

Question 3

The following function **magicfun( )** is a part of some class. What will the function magicfun( ) return, when the value of **n=7** and **n=10**, respectively? Show the dry run/working: **[5]**

```
int  magicfun( int n)
 {  if (  n= = 0)
       return   0;
    else
      return  magicfun(n/2)  * 10 + (n % 2);
 }
```

Comments of Examiners

A number of candidates answered this question correctly.
Common errors made by candidates:
the concept of recursion was not clear to some candidates;
some had problems in calling the recursive function;
the concept of LIFO (Last In First Out) was not clear and the last digit was missing in some cases.
Some candidates did not show the working and gave the answer directly.

Suggestions for teachers

- More practice should be given in solving programs using recursive techniques. Attention should be paid by teachers towards recursion and its techniques with examples.
- Knowledge of base case and recursive case should be given to students for every program using recursive technique.
- Output program using recursive technique should be given more practice.
- Memory blocks must be used to show the concept of Stack (LIFO).
- Students must be told to show the working where ever required.

MARKING SCHEME

Question 3

(i) when n=7 OUTPUT : magicfun(7)
magicfun(3) * 10 + 1
magicfun(1) *10 + 1
magicfun(0) *10 + 1
0
= 111

(ii) when n=10 OUTPUT : magicfun(10)
magicfun(5) * 10 + 0
magicfun(2) *10 + 1
magicfun(1) *10 + 0
magicfun(0) *10 + 1
0
= 1010

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 (2,3,4,5,6,7,8,10,11).

(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(P, Q, R, S) = π(0,1,2,4,5,6,8,10).

(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) Most candidates fared well in this part. Some candidates were not able to draw the K-Map for the SOP expression correctly. Different variables were used to draw the K-Map instead of those given in the question paper. For a number of candidates the "Map rolling" concept was not very clear. In some cases, redundant groups were also included in the final expression which was not required.

(ii) Most of the candidates answered correctly. Some drew the logic circuit using NAND gates while some others drew vague diagrams with different shapes instead of the standard logic gates.

(b) (i) Some candidates made errors in place value and putting variables in K-Map. In some cases the groups were reduced by laws. A few candidates drew the K-Map incorrectly. Several candidates included the redundant group in the final expression.

(ii) A number of candidates drew the logic circuit using NOR gates while some others drew vague diagrams.

Suggestions for teachers

- Make students reduce SOP and POS expressions using K-Map simultaneously. Students should be told not to include the redundant group in the final expression. Practice should be given in drawing the K-Map, filling the K-Map with 0's and 1's, marking the groups and reducing the groups.
- More practice should be given in drawing logic circuits using basic gates and also with universal gates.
- Emphasize on arranging the variables in proper order and the importance of cell values corresponding with the variables. Explain clearly how the groups are framed and reduced. Redundant groups are not to be included in the final reduced expression.

MARKING SCHEME

Question 4

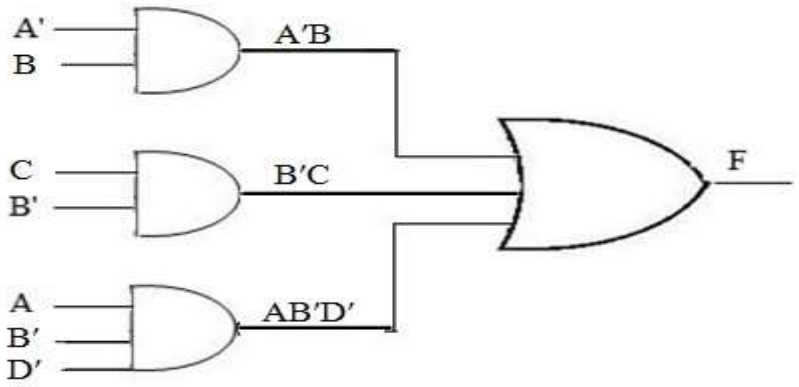
(a) $F(A,B,C,D) = \sum (2 , 3 , 4, 5 , 6 , 7 , 8 , 10 , 11)$

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

There are two quads and one pair:

Quad 1 ($m_{2+}\, m_{3+}\, m_{10}\, m_{11}$) = **B'C** Quad2 ($m_{4+}\, m_{5+}\, m_{6+}\, m_{7}$) = **A'B**
Pair ($m_{8+}\, m_{10}$) = **AB'D'**

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



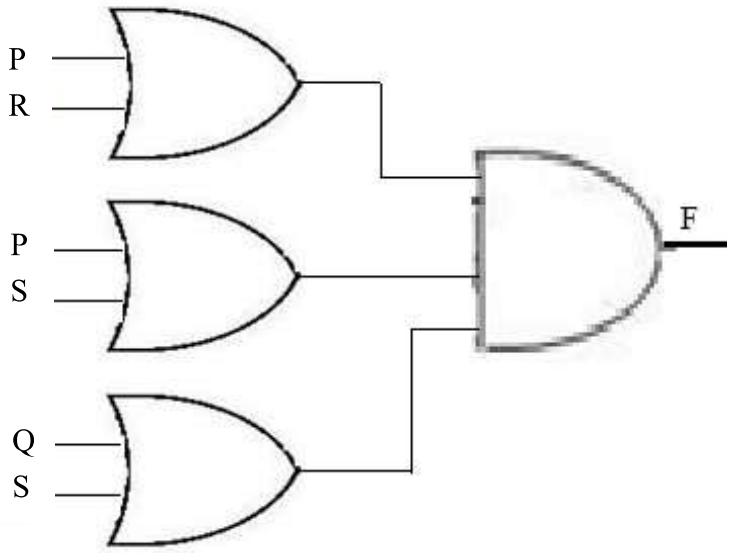
4(b) F(P,Q,R,S) = **π (0 , 1 , 2 , 4 , 5 , 6 , 8 , 10)**

	R+S	R+S'	R'+S'	R'+S
P+Q	0 **0**	1 **0**	3 1	2 **0**
P+Q'	4 **0**	5 **0**	7 1	6 **0**
P'+Q'	12 1	13 1	15 1	14 1
P'+Q	8 **0**	9 1	11 1	10 **0**

There are three quads :

Quad 1 : ($M_0\, M_1\, M_4\, M_5$) = P + R Quad 2 : ($M_0\, M_2\, M_4\, M_6$) = P + S
Quad 3 : ($M_0\, M_2\, M_8\, M_{10}$) = Q + S

Hence F(P,Q,R,S) = (P + R) . (P + S) . (Q + S)



Question 5

(a) A school intends to select candidates for an Inter-School Essay Competition as per the criteria given below: **[5]**

- The student has participated in an earlier competition and is very creative.

OR

- The student is very creative and has excellent general awareness, but has not participated in any competition earlier.

OR

- The student has excellent general awareness and has won prize in an inter-house competition.

The inputs are:

INPUTS	
A	participated in a competition earlier
B	is very creative
C	won prize in an inter-house competition
D	has excellent general awareness

(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 **POS** expression for **X(A,B,C,D)**.

(b) State the application of a *Half Adder*. Draw the truth table and circuit diagram for a Half Adder. **[3]**

(c) Convert the following Boolean expression into its canonical POS form: **[2]**

$$F(A,B,C) = (B + C') \cdot (A' + B)$$

Comments of Examiners

(a) While a number of candidates answered this part well, some did not mention the final expression. Several candidates were confused with the POS expression and took the output with 1's instead of 0's. Some reduced the expression using K-Map which was not required.

(b) Some candidates drew the block diagram while some others drew the Full adder instead of Half adder. The truth table and logic circuit for the 'Partial sum' and 'Carry' were interchanged in a few cases.

(c) Candidates used various methods to convert the expression. Some were not clear with the term canonical. Working/steps were not shown in many cases.

Suggestions for teachers

- Truth table with 4 input variables (i.e. 16 combinations) must be given for practice. Propositional logic must be explained to find the criteria for the output. Candidates should be told to write the final expression in either Canonical or Cardinal form for both SOP and POS expressions.
- More practice should be given so that students know the circuit diagram, truth table, expression, definition and use for all applications of Boolean algebra i.e. Half adder, Full adder, Encoders, Decoders, etc.
- Boolean expression with SOP and POS must be practiced in both canonical and cardinal form along with their differences. Their inter-conversion must also be practiced.

MARKING SCHEME

Question 5

(a)

A	B	C	D	X (OUTPUT)
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	1
0	1	0	0	**0**
0	1	0	1	1
0	1	1	0	**0**
0	1	1	1	1
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

POS Expression: X (A ,B, C, D) = π (0,1, 2, 4, 6, 8, 9, 10)

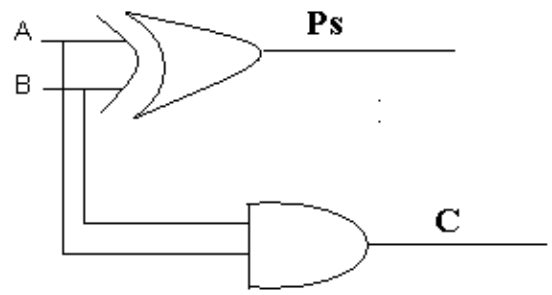
X(A ,B, C, D) = (A+B+C+D) . (A+B+C+D') . (A+B+C'+D) . (A+B'+C+D) . (A+B'+C'+D) .(A'+B+C+D) . (A'+B+C+D') . (A'+B+C'+D)

(b) Application of Half Adder is to perform partial addition of two bits.

Truth table of Half Adder :

A	B	P_S	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit diagram for a Half Adder :



(c) Convert to Canonical form: F(A,B,C)=(B+C') . (A'+B)
= (B+C'+0) . (A'+B+0)
= (B+C'+(A.A')) . (A'+B+(C.C'))
= (A+B+C') . (A'+B+C') . (A'+B+C)

Question 6

(a) What is a *Multiplexer?* How is it different from a *decoder?* Draw the circuit diagram for a 8:1 Multiplexer. **[5]**

(b) Prove the Boolean expression using Boolean laws. Also, mention the law used at each step. **[3]**

$$F = (x' + z) + [\ (y' + z) \bullet (x' + y)\ ]' = 1$$

(c) Define *maxterms* and *minterms*. Find the maxterm and minterm when: **[2]**

P = 0, Q = 1, R = 1 and S = 0

Comments of Examiners

(a) The definition was well answered by most of the candidates. Some were confused with logic diagram of a multiplexer and drew the diagram of a decoder instead. Some used the OR gate instead of AND gate in the logic diagram. A few candidates drew the block diagram for a multiplexer. In some cases, the input signals were missing, while some others drew the 4:1 MUX instead of 8:1 MUX.

(b) A number of candidates did not mention the laws. Some used lengthy methods for reducing and wasted their time. Others used the algebraic method to prove.

(c) A number of candidates did not give a proper definition. Finding the *maxterm* and *minterm* was well attempted by many candidates but some interchanged the answers. A few candidates wrote the definition of SOP and POS instead of *maxterms* and *minterms.*

Suggestions for teachers

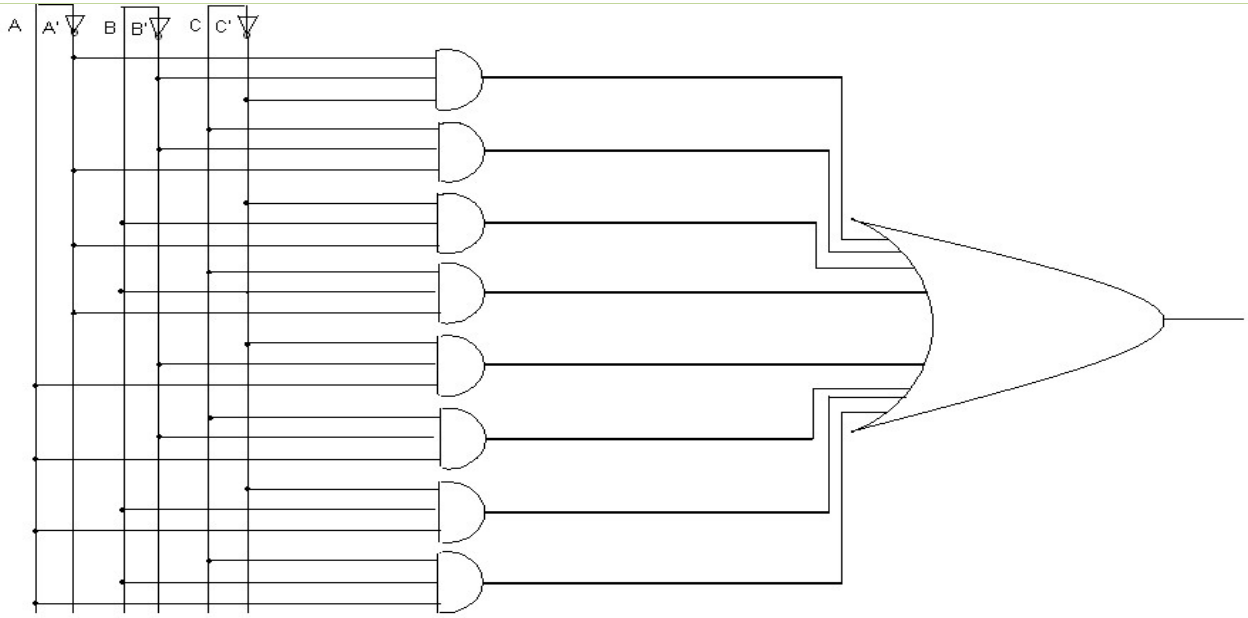
- Most practice should be given in drawing Multiplexer, Encoder and Decoder. Use of proper connector and gates must be explained. Uses for each application along with their differences should be clarified.
- More practice must be given in reducing expression using laws. Mention of laws while reducing the expression must be encouraged. More practice must also be given in L.H.S. = R.H.S. type of questions in Boolean algebra.
- Students must be told to read the question properly and answer accordingly.

MARKING SCHEME

Question 6

(a) A **Multiplexer** is a combinational circuit which inputs parallel data and outputs one serial data where as, a **Decoder** is a combinational circuit which inputs n lines and outputs 2^n or fewer lines.

Circuit diagram of a 8:1 Multiplexer :



(b) Prove: $F = (x'+z) + [\,(y'+z).(x'+y)\,]' = 1$

$= (x'+z) + (y'+z)' + (x'+y)'$ **(De Morgan Law)**

$= x'+z + y.z' + xy'$ **(Distributive Law)**

$= (x'+x)(x'+y') + (z+z')(z+y)$

$= x'+y'+z+y$

$= 1$ (as $y+y'=1$) **(Identity Law)**

(c) **Maxterm :** It is the sum of all its literals.
Minterms : It is the product of all its literals.

When P= 0 , Q = 1 , R = 1 , S = 0

Maxterm = P + Q' + R' + S
Minterms = P' Q R S'

SECTION – B

*Answer **any two** questions.*

Question 7

A class **Palin** has been defined to check whether a positive number is a *Palindrome* number or not. **[10]**

The number 'N' is palindrome if the original number and its reverse are same.

Some of the members of the class are given below:

Class name	:	**Palin**
Data members/instance variables:		
num	:	integer to store the number
revnum	:	integer to store the reverse of the number
Methods/Member functions:		
Palin( )	:	constructor to initialize data members with legal initial values
void accept( )	:	to accept the number
int reverse(int y)	:	reverses the parameterized argument 'y' and stores it in **'revnum'** using **recursive technique**
void check( )	:	checks whether the number is a Palindrome by invoking the function **reverse( )** and display the result with an appropriate message

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

Comments of Examiners

Additional instance variables were used by a number of candidates. Instance variables were declared again in the constructor by some of the candidates. The concept of recursion was not clear to many candidates. Some did not use the parameters in the function reverse( ), others wrote the function reverse( ) without using the recursive technique. Several candidates had problems in check( ) function, where some candidates did not invoked the reverse( ) function. The other function including the constructor was well answered. Object creation and method calling was not done properly in the main( ) function in some cases. A few candidates did not write the main( ) function.

Suggestions for teachers

- More practice should be given to solve programs using recursive techniques.
- Knowledge of base case and recursive case should be given to the students for every program using recursive technique.
- Invoking function within another function should be given more practice.
- The difference between iteration and recursion must be explained.
- Knowledge of instance variables and their accessibility in the class must be emphasized.
- Candidates must be advised to read the question and answer accordingly what is required.

MARKING SCHEME

Question 7

```
import java.util.*;
public class Palin
{
   int num,revnum;
   static Scanner x=new Scanner(System.in);
   Palin()
   { num=0;revnum=0;
   }
   void accept()
   { System.out.println( "Enter a number");

     num=x.nextInt();
   }
   int reverse(int y)
   {
     if(y>0)
     { revnum =revnum * 10 + y%10;
      return reverse(y/10);
     }
     else
     return revnum;
   }
   void check()
   {
     int p=num;
     if( num==reverse(p))
       System.out.println("palindrome");
     else
       System.out.println("not a palindrome");
   }
   static void main()
   {
     Palin obj=new Palin();
     obj.accept();
     obj.check();
   }
}
```

Question 8

A class **Adder** has been defined to add any two accepted time. **[10]**

Example: Time A - 6 hours 35 minutes
Time B - 7 hours 45 minutes
Their sum is - 14 hours 20 minutes (where 60 minutes = 1 hour)

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

[Class name	:	**Adder**
Data member/instance variable:		
a[ ]	:	integer array to hold two elements (hours and minutes)
Member functions/methods:		
Adder( )	:	constructor to assign 0 to the array elements
void readtime( )	:	to enter the elements of the array
void addtime(Adder X, Adder Y)	:	adds the time of the two parameterized objects X and Y and stores the sum in the current calling object
void disptime( )	:	displays the array elements with an appropriate message (i.e. hours = and minutes =)

Specify the class **Adder** giving details of the **constructor( ), void readtime( ), void addtime(Adder, Adder)** and **void disptime( ).** Define the **main( )** function to create objects and call the functions accordingly to enable the task.

Comments of Examiners

The addtime( ) function was not done properly by some candidates. Various methods/techniques were used add the time. Several candidates did it directly without using the parameterized object. A number of candidates had problem with the passing of object to the function. In some cases the candidates failed to store the sum of the two objects in the current object. Constructor and the main() method was largely answered properly.

Suggestions for teachers

- Passing of objects to a function through parameters must be given more practice.
- Working on one-dimensional and two-dimensional arrays must be explained with various examples.
- Pass by value and pass by reference must be practiced and explained in detail with examples.
- Candidates must be advised to adhere to the rubric of the question and answer accordingly.

MARKING SCHEME

Question 8

```
import java.util.*;
public class Adder
{
   int a[]=new int[2];
   static Scanner x=new Scanner(System.in);
   Adder()
   { a[0]=0;a[1]=0;
   }
   void readtime()
   { System.out.println("Enter hours and minutes");
     a[0]=x.nextInt();
     a[1]=x.nextInt();
   }
   void disptime()
   {
      System.out.println("Hours=" + a[0]);
      System.out.println("Minutes=" + a[1]);
   }
   void addtime(Adder X,Adder Y)
   { a[1]=X.a[1] + Y.a[1];
     a[0]=a[1]/60;
     a[1]=a[1]%60;
     a[0] += X.a[0] + Y.a[0];
   }
   static void main()
   { Adder a=new Adder();
     Adder b=new Adder();
     Adder c=new Adder();
     a.readtime();
     b.readtime();
     c.addtime(a,b);
     c.disptime();
   }

}
```

Question 9

A class **SwapSort** has been defined to perform string related operations on a word input. **[10]**

Some of the members of the class are as follows:

Class name	:	**SwapSort**
Data members/instance variables:		
wrd	:	to store a word
len	:	integer to store length of the word
swapwrd	:	to store the swapped word
sortwrd	:	to store the sorted word
Member functions/methods:		
SwapSort( )	:	default constructor to initialize data members with legal initial values
void readword( )	:	to accept a word in UPPER CASE
void swapchar( )	:	to interchange/swap the first and last characters of the word in '**wrd**' and stores the new word in '**swapwrd**'
void sortword( )	:	sorts the characters of the original word in alphabetical order and stores it in '**sortwrd**'
void display( )	:	displays the original word, swapped word and the sorted word

Specify the class **SwapSort**, giving the details of the **constructor( ), void readword( ), void swapchar( ), void sortword( )** and **void display( )**. Define the **main( )** function to create an object and call the functions accordingly to enable the task.

Comments of Examiners

Different methods / logic were used swap characters in swapchar( ) function and to sort in alphabetical in sortword( ) function. Some candidates included local variables in the functions and shared it with other functions. Others used the Character array to sort the word instead of doing it directly. In a few cases, the replace function was used which was not required. Some candidates were confused in extracting the middle characters in swapchar( ) function. A number of candidates were not able to display the required output in the display( ) function. The main( ) function and constructor were not answered properly by some of the candidates.

Suggestions for teachers

- Practice should be given in extracting characters from words, words from sentences and sentences from paragraphs. Different methods /logic should be adopted so that wider exposure to string manipulation related programs is given to students.
- Knowledge of constructors to initialize a string and other data members should be given.
- Conversion of string into characters and concatenating of strings must be given more practice.

MARKING SCHEME

Question 9

```
import java.util.*;
public class SwapSort
{   String wrd,swapwrd,sortwrd;
    int len;
    static Scanner x=new Scanner(System.in);
    SwapSort()
    {
       swapwrd="";
       sortwrd="";
    }
    void readword()
    {
       System.out.println("Enter word in Upper case");
       wrd=x.next();
       len=wrd.length();
    }
    void swapchar()
    { swapwrd=wrd.charAt(len-1) + wrd.substring(1,len-1) + wrd.charAt(0);
    }
    void sortword()
    {    char c;
       for(int i=65;i<=90;i++)
       { for(int j=0;j<len;j++)
          { c=wrd.charAt(j);
            if(c==i)
              sortwrd += c;
          }
       }
    }
    void display()
    {
      System.out.println("Original word = " + wrd);
      System.out.println("Swapped word = " + swapwrd);
      System.out.println("Sorted word = " + sortwrd);
    }
    static void main()
    { SwapSort x=new SwapSort();
     x.readword();
     x.swapchar();
     x.sortword();
     x.display();
    }
}
```

SECTION – C

*Answer **any two** questions.*

Question 10

A *super class* **Product** has been defined to store the details of a product sold by a wholesaler to a retailer. Define a *sub class* **Sales** to compute the total amount paid by the retailer *with* or *without fine* along with *service tax*. **[5]**

Some of the members of both the classes are given below:

Class name	:	**Product**
Data member/instance variable:		
name	:	stores the name of the product
code	:	integer to store the product code
amount	:	stores the total sale amount of the product (in decimals)
Member functions/methods:		
Product(String n, int c, double p)	:	parameterized constructor to assign data members name=n, code=c and amount = p
void show( )	:	displays the details of the data members
Class name:		**Sales**
Data member/instance variable:		
day	:	stores number of days taken to pay the sale amount
tax	:	to store the service tax (in decimals)
totamt	:	to store the total amount (in decimals)
Member functions/methods:		
Sales(…)	:	parameterized constructor to assign values to data members of both the classes
void compute( )	:	calculates the service tax @ 12·4% of the actual sale amount calculates the fine @ 2·5% of the actual sale amount **only** if the amount paid by the retailer to the wholesaler exceeds 30 days calculates the total amount paid by the retailer as (actual sale amount + service tax + fine)
void show( )	:	displays the data members of super class and the total amount

*Assume that the super class **Product** has been defined*. Using the **concept of inheritance**, specify the class **Sales** giving the details of the **constructor(…)**,**void compute( )** and **void show( )**.

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

Comments of Examiners

The concept of Inheritance was not clear to many candidates. The keywords 'extends' and 'super' were missing in some cases. Constructor with inheritance was not answered correctly. Accessing the members of the super class by the derived class was not clear to a number of candidates.

Some candidates declared the base class also, which was not required. Data members were not declared properly by some candidates. The function compute() in the derived class was not answered properly. In some cases, algorithm was written instead of a program. The rest of the function were well answered.

Suggestions for teachers

- Practice should be given to students on inheritance. The importance of the keywords 'extends' and 'super' in inheritance must be explained properly. Use of constructor of the base class should be made clear.
- Explain the different visibility modes and their accessing capability.
- Calling the member function from the super class to the derived class must be made clear.
- Instruct students to read the question properly (base class not required) and answer accordingly.
- The concept of overriding in inheritance must be explained with examples.

MARKING SCHEME

Question 10

```
public class Sales extends Product
{  int day;
   double tax,totamt;
   Sales( String n,int a, double b, int d)
      {  super(n,a,b);
         day=d;
      }
    void compute()
      {   double f=0.0;
          tax= (12.4 /100) * amount;
          if(day>27)
          f=(2.5/100)* amount;
         totamt= amount+tax+f;
      }
   void show()
      {   super.show();
          System.out.println("No of days=" + day);
          System.out.println("Sales Tax=" + tax);
         System.out.println("Total Amount=" + totamt );
      }
  }
```

Question 11

Queue is an entity which can hold a maximum of 100 integers. The queue enables the user to add integers from the rear and remove integers from the front. **[5]**

Define a class **Queue** with the following details:

Class name	**:**	**Queue**
Data Members / instance variables:		
Que[ ]	:	array to hold the integer elements
size	:	stores the size of the array
front	:	to point the index of the front
rear	:	to point the index of the rear
Member functions:		
Queue (int mm)		constructor to initialize the data size = mm, front = 0, rear = 0
void addele(int v)	:	to add integer from the rear if possible else display the message "**Overflow**"
int delele( )	:	returns elements from front if present, otherwise displays the message "**Underflow**" and return -9999
void display ( )	:	displays the array elements

Specify the class **Queue** giving details of **ONLY** the functions **void addele(int)** and **int delele( )**. Assume that the other functions have been defined.

The main function and algorithm need NOT be written.

Comments of Examiners

The concept of queue was not clear to most of the candidates. Common errors made by candidates were as follows:

(i) the condition / logic for underflow and overflow was not answered correctly;

(ii) increment / decrement of front and rear index was not done properly.

The methods addele ( ) and delete( ) were found to be difficult by some of the candidates. A few candidates also defined the constructor which was not required. In some cases, the class was not defined, only the functions addele( ) and delete( ) were defined.

Suggestions for teachers

- More practice should be given in data structure programs like the stacks, queues, de queues, etc. Working must be shown as to how the stack or a queue performs (examples can be supportive).
- The concept of LIFO and FIFO must be explained to students with lively examples related to real world.
- Implementation of stacks, queues and de queues using arrays should be emphasized. Only the concept has to be explained taking the base as an array. It should be made clear to the students that it is not an array related program which can be manipulated by shifting / inserting or initializing by any value since these data structures require pointers and pointers are not supported in java.

MARKING SCHEME

Question 11

```
public class Queue
 {   int Que[]=new int[100];
     int max,f,r;
     void addele(int v)
       {  if(r<max-1)
            Que[++r]=v;
          else
            System.out.println("Overflow");
       }
    int delele()
       {  if(f!=r)
             return Que[++f];
          else
             return -9999;
       }
 }
```

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]**

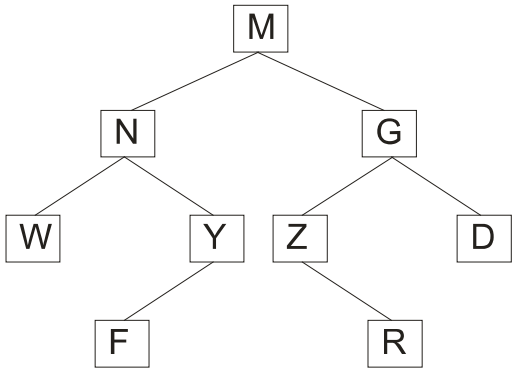
```
class   Node
{
        int     num;
        Node    next;
}
```

Write an *Algorithm* **OR** a *Method* to count the nodes that contain only odd integers from an existing linked list and returns the count.

The method declaration is as follows:

int CountOdd(Node startPtr)

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



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

(ii) State the level numbers of the nodes **N** and **R** if the root is at **0** (zero) level. **[1]**

(iii) List the internal nodes of the right sub-tree. **[1]**

Comments of Examiners

(a) Many candidates attempted this part well. Some candidates had problems in moving the pointer to the next node and checking for null. Some wrote the algorithm in simple English language, covering all the main steps. In some cases, the temporary pointer was not created.

(b) (i) Several candidates wrote the preorder instead of post order of the tree. In some cases, one or two nodes were not placed correctly.

(ii) This part was largely answered well.

(iii) Most candidates attempted this part well. A few wrote the internal nodes of the entire tree.

Suggestions for teachers

- More methods / algorithms should be practiced with link list data structure. Use of diagrams to illustrate the link list must be practiced.
- Knowledge of temporary pointer, checking for null condition and moving pointer to the next node must be given.
- Root, height, depth, size, degree, siblings, nodes (internal and external), levels, tree traversals, etc. must be explained 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. Return count
Step 4. Check for odd and increment the counter.
Step 5. Move pointer to the next node
Step 6. End

METHOD:

```
int CountOdd(Node startPtr)
  {  int c=0;
     Node temp=new Node(startPtr);
     while(temp != null)
        { if (temp.num  % 2 != 0)
             c++;
          temp=temp.next;
         }
      return c;
   }
```

(b) (i) W F Y N R Z D G M

(ii) Level of N=1 and Level of R=3

(iii) G and Z

GENERAL COMMENTS

Topics found difficult by candidates

- The symbols ' =>' , 'ʌ' and 'v' from propositional logic (Inverse and Converse)
- Interfaces and Classes
- Complexity
- Returning value of the base case in recursive output
- K-MAPS (Grouping, map-rolling, place value)
- Complement properties
- Recursive technique
- Passing objects to functions
- Queue operations for adding and removing elements

Concepts in which candidates got confused

- The symbols in a proposition
- The terms 'complexity' and 'interface'
- Output using recursive technique
- Passing objects to functions
- Sorting and swapping character in a word
- Passing of objects
- Use of Single instance variable for multiple operations in various functions
- Link list and Queues

Suggestions for candidates

- Prepare summary for each chapter or use high lighters to recognize the important terms and definitions.
- Practical work on the system on a regular basis is necessary to understand the syntax and to correct errors.
- Answers and definitions should be short and precise and according to marks intended.
- Working should be shown at the side of each question wherever required.
- Laws must be mentioned while reducing a Boolean Expression.
- Practice one form of K-Map with proper place value for both SOP and POS.
- In programming, documentation is compulsory 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.
- Do not memorize the program, try to understand the logic. Practice constructors with every program.
- Treat each function of a class as separate program.