ISC

COMPUTER SCIENCE

PAPER 1 (THEORY)

- **SOLUTION OF 2010**
- **COMMENTS OF COUNCIL EXAMINERS**
- **SUGGESTIONS FOR TEACHERS**

Dedicated to all my lovely students. May God help you always.

This small booklet contains solution of 2010 ISC Computer Science Paper 1 (Theory). The comments from the council examiners under solution of every question makes this a very handy guide for students to understand what the council expects as answer from the students.

I hope that the students will find this to be useful.

Md. Zeeshan Akhtar

1st February, 2015

BLANK PAGE

COMPUTER SCIENCE PAPER 1 (THEORY)

PART I

*Answer **all** questions*

Question 1

(a) If X = A′ BC + AB′ C + ABC + A′ BC′ **[2]**
then find the value of X when A = 1 ; B = 0 ; C = 1

(b) Verify if. P . (~ P + Q′) = (P => Q) using truth table. **[2]**

(c) Draw the logic circuit of NOR using NAND gate only. **[2]**

(d) Convert the following function into its Canonical sum-of-products form: **[2]**

$F(X,Y,Z) = \sum (0,1,5,7)$.

(e) Show that dual of P′ QR′ + PQ′R + P′ Q′ R is equal to the complement of: **[2]**

PQ′ R + Q. (P′ R′ + PR′)

Comments of Examiners

(a) A few candidates were confused with the term decoding and calculated the decimal equivalent of the expression. Some drew the complete truth-table.

(b) Some candidates were confused with the basic symbols (~) and (=>) of propositional logic. Most of the candidates answered this part of the question correctly.

(c) Drawing logic circuits with NOR gates were not clear in some cases. Some drew the circuit using AND and OR gates. In some cases incomplete circuit was drawn.

(d) The concept of generating maxterms and minterms and converting the Canonical form to Cardinal form was not answered properly. Some used the Truth Table to solve the problem. Most of the candidates answered this part of the question well.

(e) Some candidates were confused with the concept of Duality with Complementation. Some reduced the expression and then found the duality.

Suggestions for teachers

- Difference between binary addition and postulates should be made clear to the students.
- Proper practice of propositional logic and the different types of operators along with their uses should be explained to the students in detail.
- Drawing logic circuits with basic gates (AND OR, NOT) and also with universal gates (NAND, NOR) should be given more practice.
- Term Encoding and Decoding should be given more practice with SOP and POS expressions / terms. Canonical and Cardinal form of expression must be taught in detail to the students. Concept of maxterms and minterms should be given more practice.
- Difference between Duality and Complementation should be explained to students with examples.

MARKING SCHEME

Question 1.

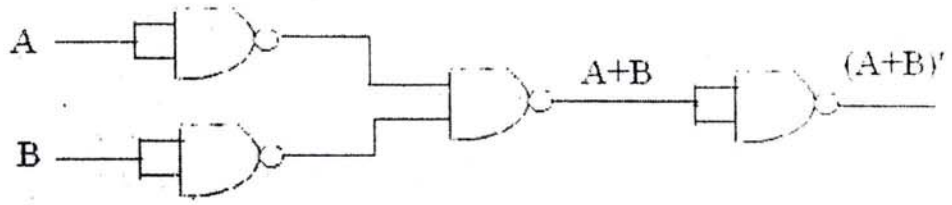
(a) X = A'BC + AB'C + ABC + A'BC'
= 1'.0.1 + 1.0'.1 + 1.0.1 + 1'.0.1'
= 0 + 1 + 0 + 0
= **1**

(b)

P	Q	~P	Q'	~P+Q'	P.(~P+Q')	P =>Q
0	0	1	1	1	0	1
0	1	1	0	1	0	1
1	0	0	1	1	1	0
1	1	0	0	0	0	1

[Not verified]

(c) NOR = (A + B)' = A ' . B '



(d) 0 = X'.Y'.Z' ; 1 = X'.Y'.Z ; 5 = X.Y'.Z ; 7 = X.Y.Z
= X'Y'Z' + X'Y'Z + XY'Z + XYZ

(e) Dual of **P'QR' + PQ'R + P'Q'R**
= (P'+Q+R') . (P+Q'+R) . (P'+Q'+R)

Complement of **PQ'R + Q. (P'R' + PR')**
= [PQ'R + P'QR' + PQR']'
= (PQ'R)' . (P'QR')' . (PQR')'
= (P'+Q+R') . (P+Q'+R) . (P'+Q'+R) .

Question 2

(a) State the difference between an Interface and a Class. **[2]**

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

(A + B) / C * (D + E)

(c) A character array B[7] [6] has a base address 1046 at 0,0. Calculate the address at B[2][3] if the array is stored **Column Major** wise. Each character requires two bytes of storage. **[2]**

(d) State the use of exceptional handling. Name the two types of exceptions. **[2]**

(e) (i) What is the worst–case complexity of the following code segment: **[1]**

```
for (int i=0 ; i < N ; i++)
  { sequence of statements
  }
for (int j=0 ; j < M ; j++)
  { sequence of statements
  }
```

(ii) How would the complexity change if the second loop went to N instead of M? **[1]**

Comments of Examiners

(a) The 'Class' definition was answered correctly by most of the candidates where as only a few candidates could give the correct definition of an 'Interface'. Examples were used in some cases to show the differences.

(b) Most candidates were able to solve this problem correctly. Some candidates had written the correct answer without showing the working. Some had applied the postfix for the brackets correctly, but could not derive the final answer.

(c) Most of the candidates had written the formula correctly, but could not substitute the values in the formula. Some candidates wrote the answer correctly without showing the working

(d) Most of the candidates could answer only the use of exceptional handling but could not name the types of exceptions. Vague answers were given for exception handling, but examples were given correctly.

(e) Concept of complexities was not clear to most of the candidates as it is a new introduction to the syllabus. Candidates had problem in calculating the complexity of the code segment. Candidates have either scored full credit or not scored at all.

Suggestions for teachers

- Proper relation between an Interface, Class and Abstract class should be explained to the students using simple examples along with their definitions.
- Examples need to be practiced with conversion of Infix to Postfix notation along with the order of precedence. The stack method should also be explained to students.
- Practice should be given in understanding the formula using Row major and Column major and to substitute the values accordingly. The cell diagram method can also be explained especially when the array size is small.
- Exception Handling should be practiced in programming and the two categories of exception should be explained with examples of each category.

- More practice should be given to the students, as 'Complexity' is a new topic. Notes should be provided and explained in detail with the concept of loops, statements etc.

MARKING SCHEME

Question 2.

(a) **Interface:-** Interface defines only abstract methods and final fields with constant value, i.e. defines a protocol of behavior.

Class :- A class is a user defined data type and is a collection of methods which are defined within an encapsulation. Class cannot have multiple inheritance.

(b) **(A + B) / C * (D + E)**

AB+ / C * DE+

AB+C/ * DE+

AB+C/DE+*

Stack	Expression
((	A
((+	AB
(	AB+
(/	AB+C
(*	AB+C/
(*(+	AB+C/DE
EMPTY	AB+C/DE+*

(c) Column major address formula: $B[p][q] = BA + W (m (q - L_1) + (p - L_2))$

$W = 2$, $B[p][q] = ?$, $L_1 = 0$, $L_2 = 0$, $p = 2$, $q = 3$, $m=7$(row), $BA = 1046$

$B[2][3] = 1046 + 2 (3 * 7 + 2)$

$= 1046 + 46 =$ **1092** Ans.

(d) Exception Handling is a transparent and neat way for handling program errors.

Two types are: checked and unchecked exceptions.

(e) Complexity – (i) O(N+M)

(ii) O(N+N) = O (2N) or O(N) when constant is dropped.

Question 3

(a) The following functions **numbers (int)** and **numbers1 (int)** are a part of some class. Answer the questions given below showing the dry run/working:

```
public void numbers(int n)
{  if (n > 0)
   {
      System.out.print(n + " ");
      numbers(n-2);
      System.out.print(n + " ");
   }
}
public String numbers1(int n)
{   if (n <= 0)
      return "";
      return numbers1(n-1) + n + " ";
}
```

(i) What will be the output of the function **numbers** (int n) when n = 5 ? **[2]**

(ii) What will the function **numbers1** (int n) return when n = 6 ? **[2]**

(iii) State in one line what is the function **numbers1**(int) doing apart from recursion? **[1]**

(b) The following function is a part of some class. It sorts the array a[ ] in ascending order using insertion sort technique. There are some places in the code marked by **?1? , ?2? , ?3? , ?4? , ?5?** which must be replaced by expression / statement so that the function works correctly.

```
void insertsort (int a [ ] )
{
   int m = ?1? ;
   int b,i,t ;
    for (i = ?2? ; i <m ; i++)
      {
         t = a[ i ] ;
         b = i - 1 ;
         while (?3? >= 0  &&  t < a [ b ] )
            {
               a[b+1] = a[b];
                ?4? ;
            }
               ?5? = t;
      }
}
```

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

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

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

(iv) What is the expression or statement at ?4? **[1]**

(v) What is the expression or statement at ?5? **[1]**

Comments of Examiners

(a) Most of the candidates answered this question correctly. Some candidates were not clear with the concept of recursive technique. Some did not show the working / dry run. Some candidates gave incomplete answers. The concept of returning string in a recursive function was not clear in some cases.

(b) Only a few candidates managed to get full credit for answering well. A number of candidates were not clear with Insertion Sort technique and hence could not answer all the parts of the question correctly. Some had problem in finding the size of the array and checking the conditions.

Suggestions for teachers

- More practice should be given to students on program using recursive technique.
- The concept of Base case and Recursive case should be explained to students with each recursive function.
- Teachers are expected to show the dry run/ working of program using the concept of stack.
- Various techniques relating to sorting and searching should be taught to the students and their differences along with their dry run must be shown and explained step wise.

MARKING SCHEME

Question 3.

(a) (i) numbers (5)
5 numbers (3)
3 numbers (1)
1 numbers (-1)

Then by LIFO 1 3 5

Ans = 531135

(ii) numbers (6)
= numbers (5) + 6 + " "
= numbers (4) + 5 + " "
= numbers (3) + 4 + " "
= numbers (2) + 3 + " "
= numbers (1) + 2 + " "
= numbers (0) + 1 + " "
" "

Ans = 1 2 3 4 5 6

(iii) Generates Natural/whole/consecutive numbers from 1 to n.

(b)	**Output :**	(i)	a . length ;
		(ii)	1 or 0
		(iii)	b
		(iv)	b - - or b = b – 1 or - - b ;
		(v)	a[b+1]

PART – II

SECTION - A

*Answer any **three** questions*

Question 4

(a) Given **F(P,Q,R,S) = Σ (0 , 2 , 5 , 7 , 8 , 10 , 11 , 13 , 14 , 15) :**

(i) Reduce the above expression by using 4 - Variable K-Map, showing the various groups (i.e octal, quads and pairs). **[4]**

(ii) Draw the Logic gate diagram of the reduced expression using NAND gate only. **[1]**

(b) Given **F(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) .**

(i) Reduce the above expression by using 4 - Variable K-Map, showing the various groups (i.e octal, quads and pairs). **[4]**

(ii) Draw the Logic gate diagram of the reduced expression using NOR gate only. **[1]**

Comments of Examiners

(a) Most candidates answered this question correctly. Some candidates made error in place value and putting variables in the K-Map. In some cases the groups were reduced by laws. Some candidates drew the K-Map incorrectly. Many candidates included the redundant group in the final expression.

(b) Most candidates fared well in this part. Some candidates were not able to draw the K-Map for the POS expression correctly. For a number of candidates the "Map rolling" concept was not very clear. The concept of cell value / place value was not clear in some cases.

Suggestions for teachers

- Instruct students to arrange the variables in proper order and the importance of cell values corresponding with the variables.
- Explain clearly how the group are formed and reduced.
- Students should be told not to include the redundant group in the final expression. Drawing circuits using universal gates should be practiced more.
- Make students practice the reduction of POS and SOP expressions using K-Map simultaneously.

MARKING SCHEME

Question 4.

(a) **F(P,Q,R,S) = Σ (0 , 2 , 5 , 7 , 8 , 10 , 11 , 13 , 14 , 15)**

(i)

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

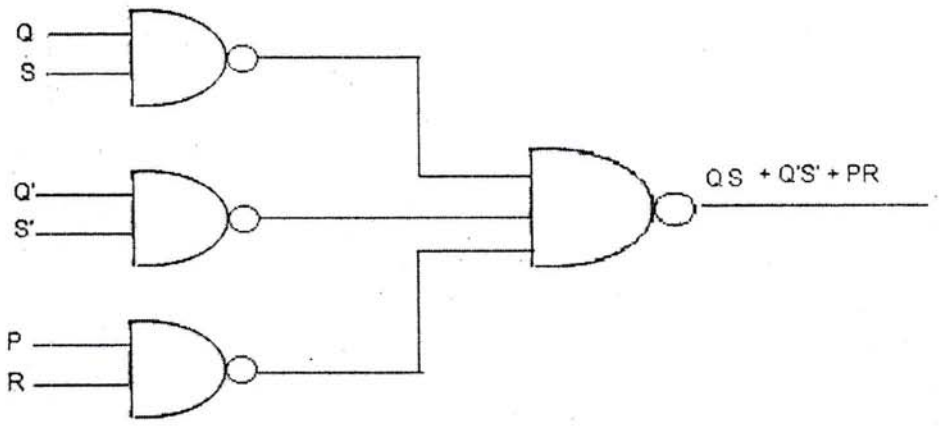
There are three quads :

Quad1 ($m_0 + m_2 + m_8 + m_{10}$) = Q'S' Quad2 ($m_5 + m_7 + m_{13} + m_{15}$) = QS

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

Hence F (P, Q, R, S) = Q'S' + QS + PR

(ii)



(b) F(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)**

(i)

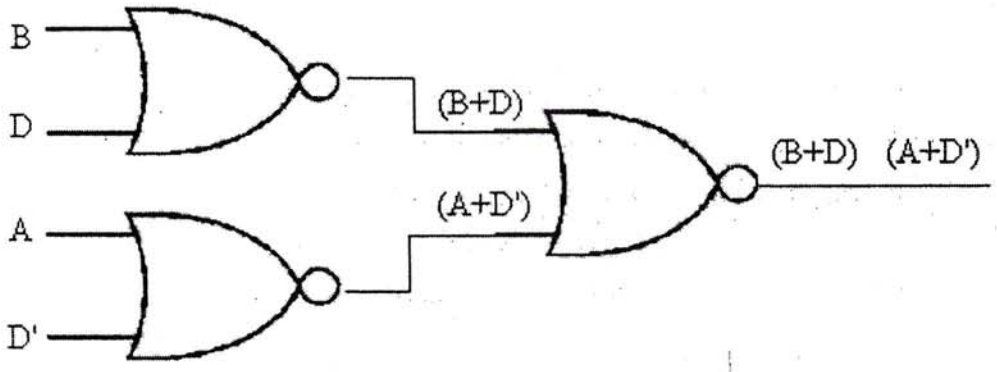
	C+D	C+D'	C'+D'	C'+D
A+B	0 **0**	1 **0**	3 **0**	2 **0**
A+B'	4 1	5 **0**	7 **0**	6 1
A'+B'	12 1	13 1	15 1	14 1
A'+B	8 **0**	9 1	11 1	10 **0**

There are two quad :

Quad 1 ($M_0 M_2 M_8 M_{10}$) = B + D Quad 2 ($M_1 M_3 M_5 M_7$) = A + D'

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

(ii)



Question 5

A Government Institution intends to award a medal to a person who qualifies any one of the following criteria:

The person should have been an Indian citizen and had lost his/her life in a war but has not completed 25 years of service.

OR

The person must be an Indian citizen and has served the nation for a continuous period of 25 years or more but has not lost his/her life in a war.

OR

The person is not an Indian citizen but has taken active part in activities for the upliftment of the nation.

The inputs are:

INPUTS	
A	The person is/was an Indian citizen
B	Has a continuous service of more than 25 years
C	Lost his/her life in a war
D	Taken part in activities for upliftment of the nation

Output: X - Denotes eligible for Medal [1 indicates YES and 0 indicates NO in all cases]

(a) Draw the truth table for the inputs and outputs given above and write the POS expression for X (A, B, C, D). [5]

(b) Reduce X (A, B, C, D) using Karnaugh's Map. [5]

Draw the logic gate diagram for the reduced POS expression for X (A, B, C , D) .

You may use gates with two or more inputs. Assume that the variable and their complements are available as inputs.

Comments of Examiners

(a) Most candidates were able to solve this question correctly. Some candidates have written incorrect input combination in the truth table and so the final expression was also wrong. Few candidates did not write the final expression. Some candidates did not read the question properly and have written the SOP expression instead of POS. Some took the Maxterms but wrote the expression in SOP form.

(b) Common errors were found in the K-Map, such as the wrong place value, marking of groups and their reduction, inclusion of redundant groups in the final expression. A few candidates drew the logic diagram without the flow lines and without labelling.

Suggestions for teachers

- More practice should be given to the students to generate truth tables using 3 variables and 4 variables . Total number of input combinations should be taught, i.e. 2^n where 'n' is the number of inputs. Deriving SOP and POS expression from a given Truth Table should be practiced more.
- Have students practice more K-Maps using pairs, quads and octets so that they understand how to take the maximum possible reduction and to make minimum number of groups.
- Teach the students the importance of flow lines and labelling while drawing the logic gate diagrams.

MARKING SCHEME

Question 5.

(a)

A	B	C	D	X
The person is/was an Indian citizen	Has a continuous service of more than 25 years	Lost his/her life in a war	Taken part in activities for upliftment of the nation	OUTPUT
0	0	0	0	0
0	0	0	1	1
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	1
1	0	1	1	1

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

X (A,B,C,D) = π (0,2,4,6,8,9,14,15)

(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)

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

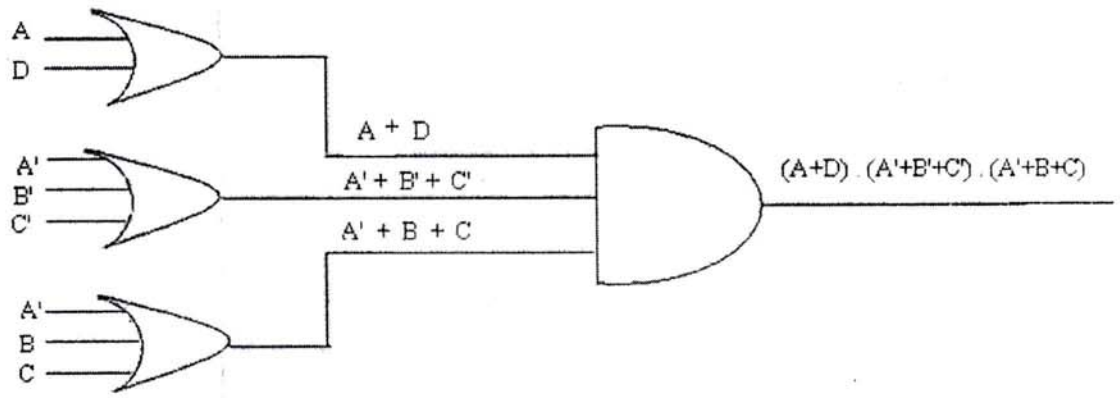
There are one quad and two pairs:

Quad 1 ($M_0 M_2 M_4 M_6$) = A + D

Pair 1 ($M_8 M_9$) = A' + B + C

Pair 2 ($M_{14} M_{15}$) = A' + B ' + C '

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



Question 6

(a) What are Maxterms? Convert the following function as a product of Maxterms: [3]

$$F(P,Q,R) = (P+Q) \cdot (P'+R')$$

(b) State whether the following expression is a Tautology or Contradiction with the help of Truth Table: [3]

$$(X <=> Z) \cdot [(X => Y) \cdot (Y => Z)]$$

(c) What is a Multiplexer? Draw the truth table and logic diagram of a 8:1 Multiplexer. [4]

Comments of Examiners

(a) Most of the candidates answered this part correctly. Some of the candidates gave vague definitions of Maxterms. The conversion of a function to its Maxterms were not clear to some candidates. Few candidates used the truth table to solve the problem.

(b) Some candidates were confused as the given expression was neither a Tautology nor a Contradiction. It was a contingency or not a total contradiction. Most of the candidates have answered the conditional (=>) and bi-conditional (<=>) columns correctly and scored full credit.

(c) Many candidates answered this question correctly. The term "Combinational Circuits" were not used in the definition by some candidates. Some drew the 4:1 multiplexer, while some drewn the block diagrams of Multiplexer instead of the logic circuit.

Suggestions for teachers

- Explain to students clearly how to convert a given expression into its Maxterm or Minterm. Also explain how to write the Canonical and Cardinal form of expression. Definitions of terms should be explained to students with proper examples.
- Give students more practice on Propositional logic and the concept of tautology, contradiction, contingency.
- Implications and bi-conditional should be explained to students.
- Encourage students to write the correct definition and also to differentiate between decoders and multiplexers along with their truth table and circuit diagrams. Also the difference between block diagram and logic circuit should be made clear to students.

MARKING SCHEME

Question 6.

(a) Maxterms are sum of all its literals.

$$F(P,Q,R) = (P+Q) . (P'+R')$$
$$= (P+Q+0) . (P'+R'+0)$$
$$= (P+Q+(RR')) . (P'+R'+(QQ'))$$

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

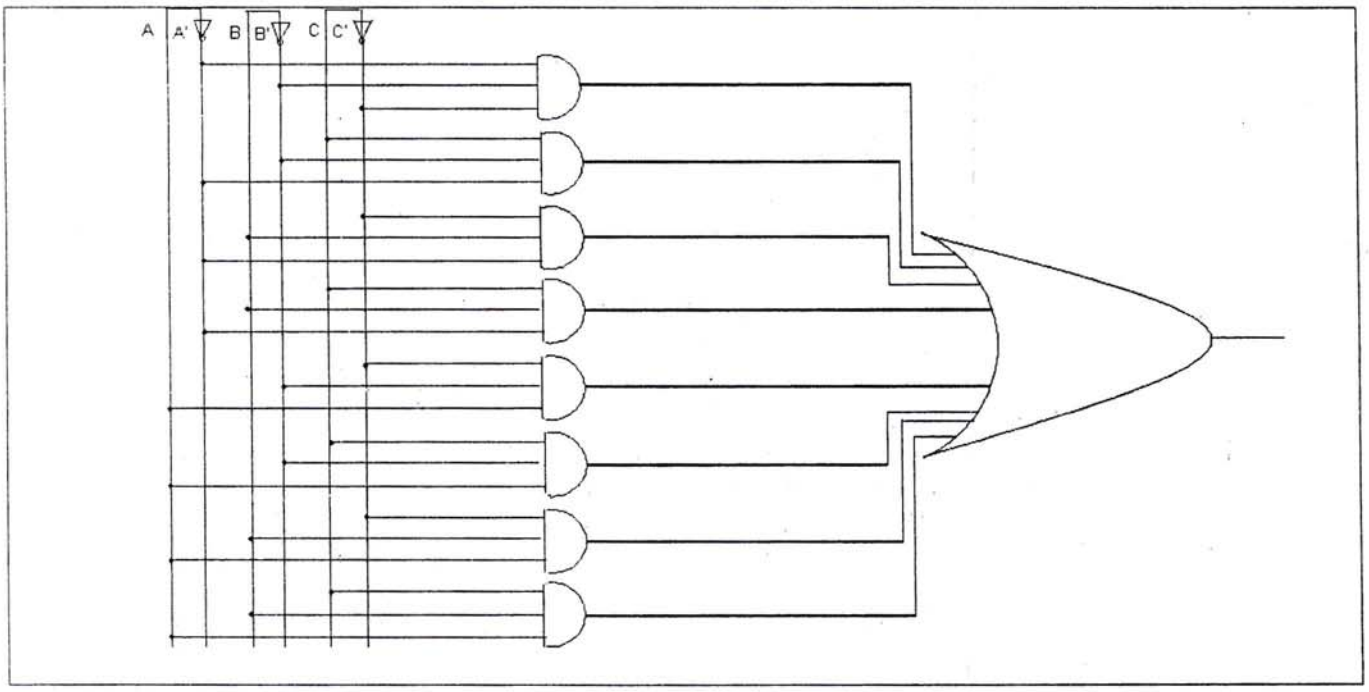
(b)

X	Y	Z	X⇔Z	X=>Y	Y=>Z	b.c	a.d
			(a)	(b)	(c)	(d)	
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	1	0	1	1	0	0	0
0	1	1	0	1	1	1	0
1	0	0	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	1	0	0	0
1	1	1	1	1	1	1	1

[It is contingency]

(c) Multiplexers are combinational circuits which inputs parallel data and outputs one serial data line. They are used for data transmission by converting parallel data into serial data .

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



Question 7

(a) Draw the circuit diagram for a 3 to 8 Decoder. [3]

(b) Draw the truth table for a Half Adder. Also derive a POS expression for the Half Adder and draw its logic circuit. [3]

(c) Simplify the following expression and also draw the circuit / gate for the reduced expression. [4]
[Show the stepwise working along with the laws used.]

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

Comments of Examiners

(a) Many candidates answered this question correctly. Some candidates used the OR gate instead of AND gate in drawing decoders. Some drew the Octal encoder instead of Octal decoder. The flow lines connecting the gates were not clear and were very untidy in some of the cases.

(b) Some candidates were confused with the POS expression of a Half Adder. The other sub parts were answered well by most of the candidates.

(c) Most of the candidates answered this part of the question correctly. Some candidates did not mention the laws and the steps for simplification were not shown. Some candidates did not draw the gate for the reduced expression.

Suggestions for teachers

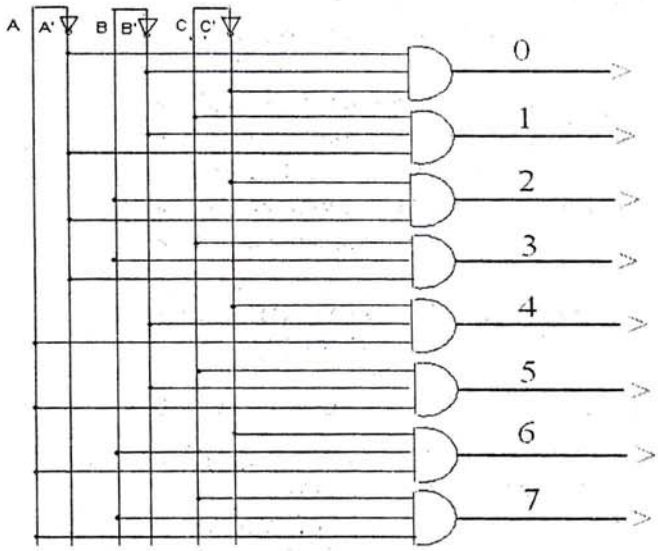
- Practice should be given to students to draw the encoder and decoder and their differences should be explained along with the truth table.
- Use of proper gate and the flow lines should be highlighted.
- More practice should be given on Half adders and Full adders along with their SOP & POS expressions, truth tables and logic circuits.

- Ask students to mention laws at every step of simplification and also show the working. Students should be asked to read the question carefully and not to overlook any sub-part of the question.

MARKING SCHEME

Question 7.

(a)



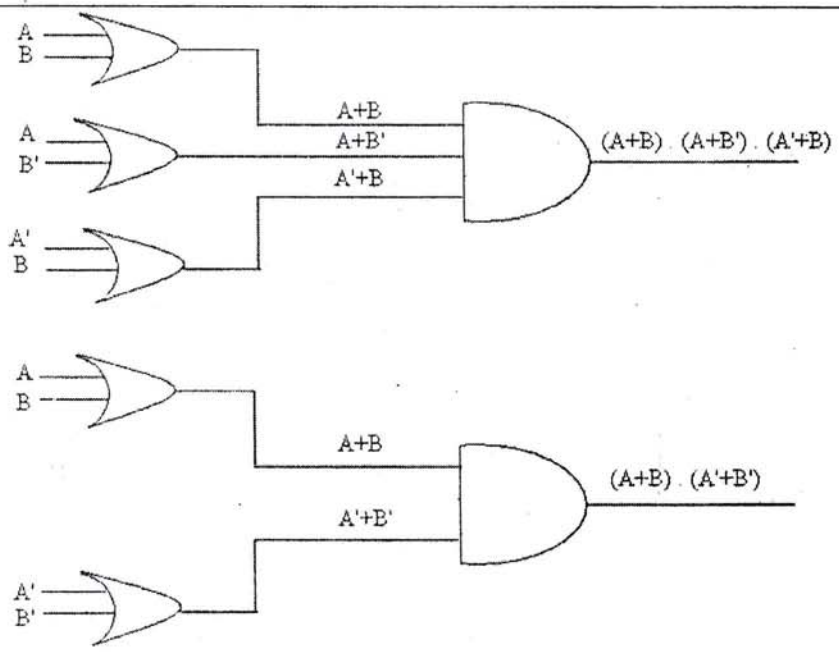
(b) Truth table of half adder :

A	B	Partial Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Partial sum (Ps) = (A+B) . (A'+B')

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

(b)



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

$= X . (Y + Z . [(X . Y)' . (X . Z)']$ (De Morgan's Law)

$= X . (Y + Z . [(X' + Y') . (X' + Z')]$

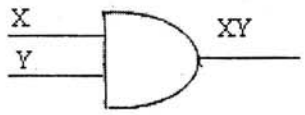
$= X . (Y + Z [X' + X' Y' + X' Z' + Y' Z'])$ (Distributive Law)

$= X . (Y + Z X' + Z X' Y' + Z X' Z' + Z Y' Z'] .)$

$= X . (Y + Z X' + Z X' Y')$ (Distributive Law)

$= XY + Z X X' + Z X' X Y)$ (Absorption Law)

$= \mathbf{XY}$ Ans.



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 8

The coordinates of a point P. on a two dimensional plane can be represented by P(x,y) with x as the x-coordinate and y as the y-coordinate. The coordinates of midpoint of two points P1(x1 , y1) and P2(x2,y2) can be calculated as P(x,y) where: **[10]**

$$x = \frac{x1+x2}{2}, \quad y = \frac{y1+y2}{2}$$

Design a class **Point** with the following details :

Class name	:	**Point**
Data Members / instance variables:		
x	:	stores the x-coordinate
y	:	stores the y-coordinate
Member functions:		
Point( )	:	constructor to initialize x = 0, y = 0
void readpoint ( )	:	accepts the coordinates x and y of a point
Point midpoint (Point A, Point B)	:	calculates and returns the midpoint of the two points A and B
void displaypoint ( )	:	displays the coordinates of a point

Specify the class **Point** giving details of the **constructor( ),** member functions **void readpoint( ), Point midpoint(Point,Point)** and **void displaypoint ( )** along with the **main( )** function to create an object and call the functions accordingly to calculate the midpoint between any two given points.

Comments of Examiners

Most of the candidates did not define the function **midpoint (point, point)** which passes and returns object to the function. Candidates were not aware how to access the data members of a class by the object through the dot (.) operator. The other functions were well answered including the constructor.

The main function was not attempted by some of the candidates. Documentation / comments were not mentioned in some cases.

Suggestions for teachers

The students should be given more practice related to passing and returning objects to the function and the concept of accessing the member of the object should be taught in more detail. Practice should be given more on different types of constructors (i.e. default, parameterized etc.). Instance variables (Global) and local variables should be explained in detail along with their scope. Students should be asked to write documentation/comments with every program as it carries marks.

MARKING SCHEME

Question 8.

```
import java.io.*;
public class Point
{
   double x, y;                                          // data members //
   BufferedReader buf = new BufferedReader (new
   InputStreamReader(System.in));
   public Point()                                        // constructor //
   {
        x = 0.0;
        y = 0.0;
    }
    public void readpoint() throws IOException           // to accept values//
    {
              System.out.println("enter x");
              x = Double.parseDouble(buf.readLine()) ;
              System.out.println("enter y");
              y = Double.parseDouble(buf.readLine()) ;
   }
   public void displaypoint()
   {
         System.out.println(x);
         System.out.println(y);
   }
   public Point midpoint(Point A, Point B)              // to calculate midpoint//
   {
```

```
        Point C = new Point() ;
        C.x = (A.x + B.x)/2;
        C.y = (A.y + B.y)/2;
        return C;
    }
    public static void main()throws IOException
    {
        Point p=new Point();                    // object creation //
        Point q=new Point();
        Point r=new Point();
        p.readpoint();
        q.readpoint();
        r=r.midpoint(p,q);
        p.displaypoint();
        q.displaypoint();
        r.displaypoint();
    }

}
```

Question 9

Input a word in uppercase and check for the position of the first occurring vowel and perform the following operation. **[10]**

(i) Words that begin with a vowel are concatenated with "**Y**".
For example, **EUROPE** becomes **EUROPEY.**

(ii) Words that contains a vowel in-between should have the first part from the position of the vowel till end, followed by the part of the string from beginning till position of the vowel and is concatenated by "**C**".
For example, **PROJECT** becomes **OJECTPRC.**

(iii) Words which do not contain a vowel are concatenated with "**N**".
For example, **SKY** becomes **SKYN.**

Design a class **Rearrange** using the description of the data members and member functions given below:

Class name : **Rearrange**

Data Members / instance variables:

Txt	:	to store a word
Cxt	:	to store the rearranged word
len	:	to store the length of the word

Member functions:

Rearrange ( )	:	constructor to initialize the instance variables
void readword ( )	:	to accept the word input in UPPER CASE
void convert ( )	:	converts the word into its changed form and stores it in string Cxt
void display ( )	:	displays the original and the changed word

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

Comments of Examiners

Most of the candidates answered this question well and scored full credit. Some candidates had problem in defining the constructor and assigning the string variable to null. The declaration of string variable, finding the position of the first occurring vowel and concatenating of two strings were incorrect in some cases. The main function was not attempted by some of the candidates. Rest of the functions were attempted and were answered correctly.

Suggestions for teachers

The declaration of string variable and its null assignment should be given more practice with programs on the computer. Proper usage of substring functions should be practiced. Knowledge of constructors should be given to the students. More programs should be practiced using string related concepts and string inbuilt functions.

MARKING SCHEME

Question 9.

```
import java.io.*;
public class Rearrange
{
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  String Txt,Cxt;
  int len;                                                    // data members //
  public  Rearrange( )
   {   Txt=null;
       Cxt=null;
        len=0;
   }
  public void readword( ) throws IOException
   {
    System.out.println(" enter word in UPPER CASE ");          // to accept a word //
    Txt = br.readLine( );
    len=Txt.length();
   }
```

```
public void convert()                                    // to convert word //
{
  int f=-1;char c;
  for( int i=0;i<len;i++)
   {
     c=Txt.charAt(i);
     if( c=='A' || c=='E' || c=='I' || c=='O' || c=='U')
      {  f=i;
         break;
      }
   }
   if(f==-1)
    Cxt=Txt.concat("N");      or   Cxt = Txt + "N";
   else if(f==0)
    Cxt=Txt.concat("Y");      or   Cxt= Txt + "Y";
   else
    {  String d=Txt.substring(f,len);
       String e=Txt.substring(0,f);
       Cxt = d+e+"C";
    }
  }
  public void display ( )
   {
     System.out.println(" Original word = " + Txt);
     System.out.println(" Changed word = " + Cxt);
   }
  public static void main( ) throws IOException
   {  Rearrange  x = new Rearrange( );
     x.readword( );
     x.convert( );
     x.display( );
   }
}
```

Question 10

Design a class **Change** to perform string related operations. The details of the class are given below:

Class name : **Change**

Data Members / instance variables:

str	:	stores the word
newstr	:	stores the changed word
len	:	store the length of the word

Member functions:

Change ( )	:	default constructor
void inputword( )	:	to accept a word
char caseconvert(char ch)	:	converts the case of the character and returns it
void recchange(int)	:	extracts characters using **recursive technique** and changes its case using caseconvert( ) and forms a new word
void display( )	:	displays both the words

(a) Specify the class **Change**, giving details of the **Constructor( )**, member functions **void inputword( ), char caseconvert(char ch), void recchange(int)** and **void display( ).** Define the **main( )** function to create an object and call the functions accordingly to enable the above change in the given word. **[8]**

(b) Differentiate between an infinite and a finite recursion. **[2]**

Comments of Examiners

(a) The declaration of string variables was incorrect in some cases. The concept of recursion was not clear to most of the candidates. Checking the Base Case in a recursive function was not properly done. Constructors and the **convert()** function were not answered properly in some cases. The main function was not written by some candidates. The rest of the functions were well answered.

(b) Some of the candidates gave vague answers for finite and infinite recursions. Some explained the differences using examples. In most cases, the term 'Base Case' was not used properly in both the definitions.

Suggestions for teachers

- Much attention should be paid by the teachers towards recursion and its techniques with examples.
- Knowledge of base case and recursive case should be given to the students for every program using recursive technique. Concept of stack memory and LIFO operations should be explained with each recursive function.
- More emphasis should be paid towards the base case and the recursive case in a recursive function, which clearly differentiates the finite with infinite recursion.

MARKING SCHEME

Question 10.

(a)
```
import java.io.*;
public class Change
{       String str, newstr;
        int  len;
        BufferedReader x = new BufferedReader(new InputStreamReader(System.in));

        public Change( ) { }                                    // default Constructor
        public void inputword( ) throws IOException
        {     System:out.println("enter a Word ");
              str = x.readLine( ) ;
              len = str.length( );
              newstr=null;
        }
        public char caseconvert(char ch)                        // to change case //
        {  if( ch>=65 && ch<=90 )          or   if( ch>='A' && ch<='Z')
                ch=(char)(ch+32);
           else if ( ch>=97  &&  ch<=122 )  or (ch>='a' && ch<='z')
                ch=(char)(ch-32);
            return ch;
        }
      public void recchange(int pos)
      {   char c;
          if (pos>-1)
            {   c= str.charAt(pos);
                recchange(pos-1);
               newstr +=caseconvert(c);
             }
        }
    public void display( )
       { System.out.println("Changed Word:" + newstr);
         System.out.println("Original Word:" + str);
        }
    public static void main ( ) throws IOException
       {  Change  x = new Change ( );
          x.inputword( );
          int p = x.len;
          x.recchange(p-1);
          x.display( );
       }
    }
```

(b) Infinite recursion – A recursive function having no base case or the base case is not reachable /attainable.

Finite recursion - A recursive function having one or more base cases and is reachable / attainable.

SECTION – C

*Answer any **two** questions*

Each Program / Algorithm should be written in such a way that it clearly depicts the logic of the problem step wise. This can also be achieved by using pseudo codes .

(Flowcharts are **not** required)

The programs must be written in Java.

The Algorithm must be written in general/standard form.

Question 11

A super class **Worker** has been defined to store the details of a worker. Define a subclass **Wages** to compute the monthly wages for the worker. The details / specifications of both the classes are given below: **[10]**

Class name	:	**Worker**
Data Members / instance variables:		
Name	:	to store the name of the worker
Basic	:	to store the basic pay in decimals
Member functions:		
Worker (...)	:	Parameterised constructor to assign values to the instance variables
void display( )	:	display the workers details
Class name	:	**Wages**
Data Members / instance variables:		
hrs	:	stores the hours worked
rate	:	stores rate per hour
wage	:	stores the overall wage of the worker
Member functions:		
Wages (...)	:	Parameterised constructor to assign values to the instance variables of both the classes
double overtime( )	:	Calculates and returns the overtime amount as (hours*rate)
void display( )	:	Calculates the wage using the formula wage = overtime amount + Basic pay and displays it along with the other details

Specify the class **Worker** giving details of the **constructor( )** and **void display ( )**. Using the **concept of inheritance**, specify the class **Wages** giving details of **constructor( ), double overtime ( )** and **void display ( )**. The **main()** function need **not** be written.

Comments of Examiners

Concept of inheritance was not clear to some candidates. Constructor with inheritance was not answered correctly. Accessing the members of the super class by the derived class was not clear to some of the candidates. Some candidates declared the super class data members as private. String data members were not declared properly by some candidates. Some did not call the **display ( )** function to display the worker's details. Some of the candidates wrote both the program and the algorithm. The rest of the functions were well answered.

Suggestions for teachers

- Practice should be given on inheritance to the students. Use of constructor using the base class member should be made clear.
- Explain the students the different visibility modes and their accessing capability. Knowledge of calling the member function from the super class to the derived class must be made clear. Use of keywords 'extends' and 'super' should be explained in detail.

MARKING SCHEME

Question 11.

```
public class Worker
  {
       protected String Name;                          // protected data members //
       protected double Basic;

        public Worker(String z, double k)              // parameterized constructor //
        {
             Name = z;
             Basic = k;
        }
       public void display( )
        {
             System.out.println("Name : " + Name);
             System.out.println("Basic Pay : " + Basic);
        }
  }
 public class Wages extends Worker                     // sub class declaration //
 {    int hrs;
      double rate, wage;

 public Wages(String a, double b, int c, double d)
 {
      super(a,b);
      hrs = c;
      rate = d;
      wage = 0.0;
 }
```

```
  public double overtime( )
  {
       return ( rate * hrs );
  }
  public void display()
  {
       double x=overtime( );
        wage = x + Super Basic;
        super.display( );
        System.out.println(" Wage = " + wage);

  }
}
```

Question 12

Define a class **Repeat** which allows the user to add elements from one end (rear) and remove elements from the other end (front) only.

The following details of the class **Repeat** are given below :

Class name	:	**Repeat**
Data Members / instance variables:		
st[ ]	:	an array to hold a maximum of 100 integer elements
cap	:	stores the capacity of the array
f	:	to point the index of the front
r	:	to point the index of the rear
Member functions:		
Repeat (int m)	:	constructor to initialize the data members cap = m, f = 0, r = 0 and to create the integer array
void pushvalue(int v)	:	to add integers from the rear index if possible else display the message("OVERFLOW")
int popvalue( )	:	to remove and return element from the front. If array is empty then return -9999
void disp ( )	:	Displays the elements present in the list

(a) Specify the class **Repeat** giving details of the **constructor(int),** member function **void pushvalue(int), int popvalue( )** and **void disp ( ).** The **main( )** function need **not** be written. [8]

(b) What is the common name of the entity described above? [1]

(c) On what principle does this entity work? [1]

Comments of Examiners

(a) Some candidates had problem in declaring the array. The constructors were not properly declared and the array was not created inside the constructor in some cases. The front and rear index were initialized to -1 instead of 0 in some cases. The functions **pushvalue()** and **popvalue()** were not answered properly. Some of the candidates wrote both the program and the algorithm. The rest of the functions were well answered.

(b) Most of the candidates answered this part of the question correctly. Some of the candidates gave vague answers. In some cases, the answer was given as Stack instead of Queue.

(c) This part of the question was well answered by most of the candidates. Some have explained with the help of examples.

Suggestions for teachers

- More practice should be given to students in data structure programs like the stacks, queues, dequeues etc. Working must be shown as to how the stack or a queue performs (examples can be supportive). Concept of front and rear index and their re-arrangement after each operation should be explained along with the checking of Overflow and Underflow conditions.
- Explain Data structures with examples to show their working and the common names that are used for these data structures.
- The concept of LIFO and FIFO must be explained to the students with lively examples related to the real world.

MARKING SCHEME

Question 12.

(a)
```
public class Repeat
{
  int st[ ] = new int[100];
  int cap, f, r;

  public Repeat ( int m)              // parameterized constructor //
  {
     cap = m;
     f = 0;
     r = 0;
     st=new int[cap];
  }
  public void pushvalue(int v)              // to add values  //
  {
     if ((r+1) <= cap )
     {  r = r+1;            or   st[++r] = v;
        st[r] = v;
     }
     else
       System.out.println( " OVERFLOW " );
  }
  public int popvalue()                          // to delete values //
  {
     int v;
     if (r != f)
     {  f = f+1;                  or   [ v = st[++f]
        v = st[f];                       return v; ]
        return v;
     }
     else
      return  -9999;
  }
  public void disp()
  {
     if ( r != f )
     {  for(int i=f+1; i<=r; i++)
        System.out.println(st[i]);
     }
     else
         System.out.println ( "  Queue is empty " );
    }
}
```

(b) The common name of the entity is **QUEUE.**

(c) It works on the principle of **FIFO.**

Question 13

(a) A linked list is formed from the objects of the class, **[4]**

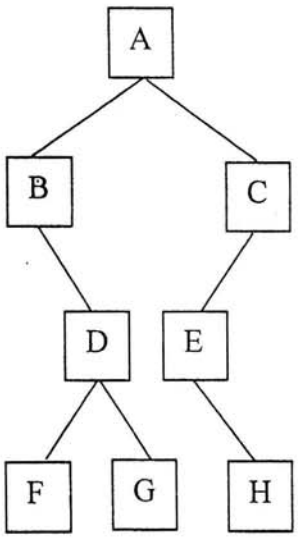
```
class ListNodes
    {
        int item;
        ListNodes next;
    }
```

Write a method **OR** an algorithm to compute and return the sum of all integers items stored in the linked list. The method declaration is specified below:

```
int listsum(ListNodes start);
```

(b) What is Big 'O' notation? State its significance. **[2]**

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



(i) Name the parent of node E. **[1]**

(ii) Write the postorder tree traversal. **[1]**

(iii) Write the internal nodes of the tree. **[1]**

(iv) State the level of the root of the tree. **[1]**

Comments of Examiners

(a) Some candidates wrote the algorithm while some wrote in program/code form. A few candidates did not initialize the accumulator variable. The null condition was not properly mentioned.

(b) Most of the candidates answered this part of the question correctly. Some of the candidates did not write the significance of Big 'O' notation.

(c) This part of the question was answered well by most of the candidates. Some sub-parts like (iv) and (v) were confusing to some of the candidates. The root was also included in the internal nodes of the tree and the level of the root was not correct in some of the cases.

Suggestions for teachers

- More programs / algorithms should be practiced with linked list and binary tree data structure. The students should be asked to write algorithms in simple language (step wise) or in standard form which clearly explains the solution of the program/problem.
- As this topic was a new introduction, teachers should provide notes and explain the various terms and significance that are used in the complexity and Big 'O' notation.
- Explain binary tree with the different parts like root, nodes (internal and external), height, depth, level, size, tree traversal (preorder, inorder and postorder), etc.

MARKING SCHEME

Question 13.

(a) **In Method Form :**

```
int listsum(ListNodes start)
  {
      int total = 0;
      ListNodes runner = start;
      while(runner != null)
        {
            total += runner.item;
            runner=runner.next;
        }
       return total;
  }
```

OR

In Algorithm Form :

Step 1. Start

Step 2. Set pointer to the starting node.

Step 3. Repeat Steps 4 and 5 until it does not reach Null or empty.

Step 4. Accumulate all the items of the node in a variable.

Step 5. Move pointer to the next node.

Step 6. Display the result

Step 7. End

(b) **Big 'O' Notation :-** Big O notation represents the performance /complexities of an algorithm and is also used for comparing the performance of two or more algorithm. It is also known as the order of calculation.

The significance of Big-O is to analyze an algorithm in terms of time complexity and space/memory /size complexity. It also provides a degree of computational complexity of an algorithm.

(c) (i) C

(ii) F G D B H E C A

(iii) B , C , D , E

(iv) 4. 0

Topics found Difficult

- Coding and decoding of SOP and POS terms.
- Propositional logic and the concept of various operators.
- To obtain the dual and complement of an expression.
- The working to convert POS to SOP form and vice versa.
- Differences between Interface and class.
- Exceptional handling, Complexity calculations and Big `O' notation.
- To derive the Maxterms and Minterms from a truth table.
- Drawing logic circuits using Universal gates (NOR , NAND).
- Place/cell values in the K- Maps and marking of Octal , Quads and pairs.
- Recursive techniques, Base case and recursive case..
- Passing objects to the function and returning objects..
- Constructor with inheritance. Use of 'extends ' and 'super ' keywords.
- String manipulation programs.
- Algorithms for linked list and other data structures.

Suggestions for Candidates

- Answers and definitions should be short and precise and according to marks allotted.
- Important words and terms should be underlined or highlighted.
- Working should be shown at the side of each question where ever required.
- Laws must be mentioned while reducing a Boolean Expression.
- Proper labeling should be done in the diagrams / logic circuits.
- 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. Practice programs on the machine.
- Treat each function of a class as a separate program.