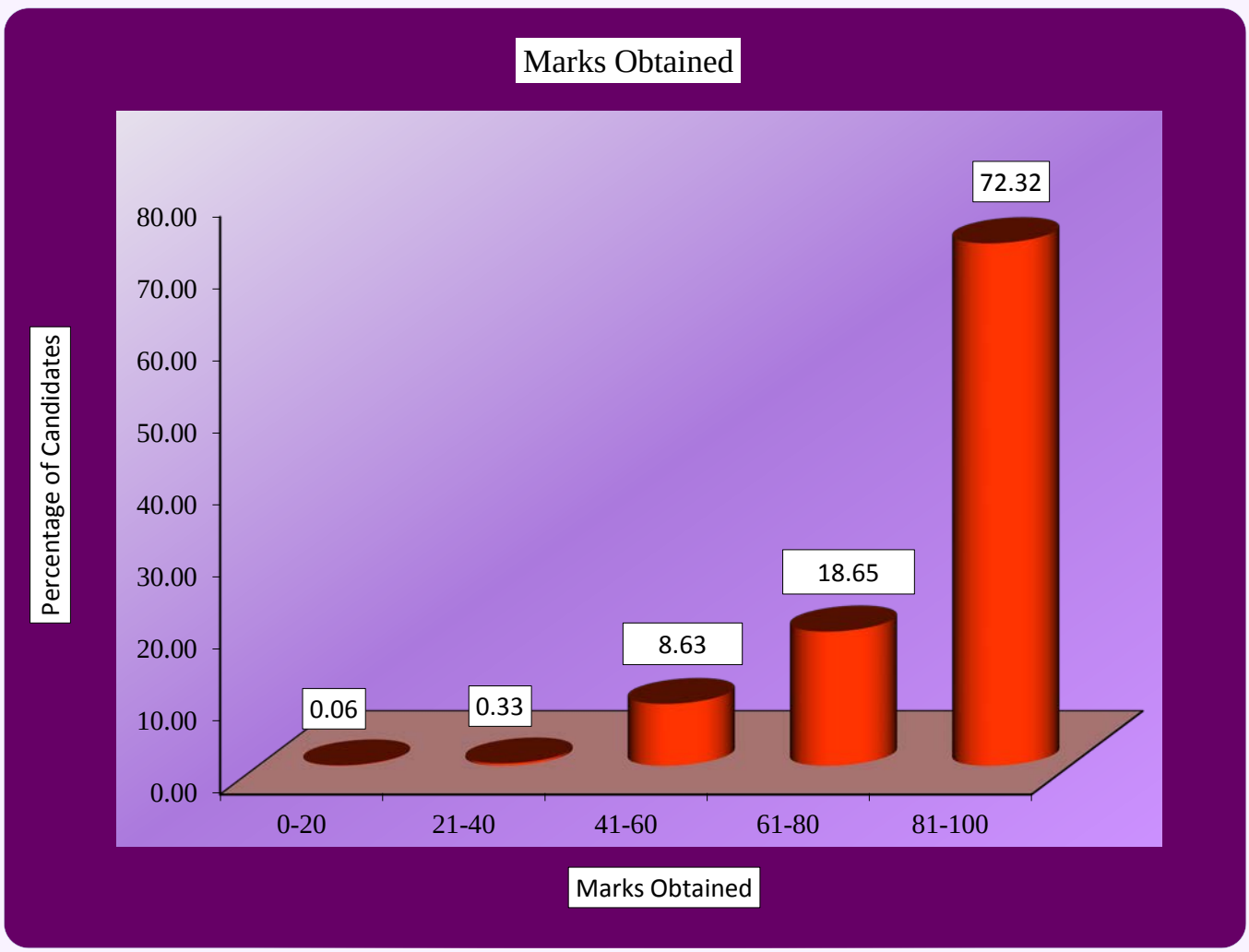
COMPUTER SCIENCE

STATISTICS AT A GLANCE

Total Number of students who took the examination	19,255
Highest Marks Obtained	100
Lowest Marks Obtained	2
Mean Marks Obtained	85.69

Percentage of Candidates according to marks obtained

Details	Mark Range				
	0-20	***21-40***	***41-60***	***61-80***	***81-100***
Number of Candidates	12	63	1662	3592	13926
Percentage of Candidates	0.06	0.33	8.63	18.65	72.32
Cumulative Number	12	75	1737	5329	19255
Cumulative Percentage	0.06	0.39	9.02	27.68	100.00



COMPUTER SCIENCE

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 Involution law and prove it with the help of a truth table. **[1]**

(b) Show that X V ~(Y Λ X) is a tautology. **[1]**

(c) Find the dual of: **[1]**

$$Y \cdot X + X' + 1 = 1$$

(d) Write the maxterm and minterm, when the inputs are A=0, B=1, C=1 and D=0. **[1]**

(e) Draw the logic circuit of a NAND gate using NOR gates only. **[1]**

Comments of Examiners

(a) This part was well answered by most of the candidates. Some candidates mentioned laws involving complementation. Others proved by laws instead of truth table. Some drew truth table of 2 variables.

(b) Most of the candidates answered this part correctly. Some candidates were confused with symbols '∧' and 'V' and interchanged them in the truth table. Some proved it with the help of Boolean laws.

(c) This was well attempted by most candidates. A few candidates did not put brackets in the dual equation. Some changed the complement also, which was not required. O's and 1's were not interchanged in some cases.

(d) Almost all candidates answered this part correctly. Some candidates interchanged the maxterms with the minterms.

(e) Some candidates were not clear about the circuit and drew vague diagrams, but using NOR gates.

Suggestions for teachers

- Give practice on all the laws of Boolean Algebra. Proving of all the laws must be emphasized.
- Teach propositional logic using all terms that are required. The symbols used in propositions must be explained.
- More practice must be given to find the dual of an equation, especially using of brackets as a order of precedence.
- Teach students to derive maxtemrs and minterms from a given truth table. Their use in expression must be explained to students.
- Logic gates and logic circuits must be practiced with almost every expression, especially with universal gates.

MARKING SCHEME

Question 1.

(a) Involution Law states that the double complement of a variable returns to the same variable.
i.e. (A')' = A
Truth Table:

A	A'	(A')'
0	1	0
1	0	1

(b) **X ∨ ~ (Y ∧ X)**

X	Y	**Y ∧ X**	**~ (Y ∧ X)**	**X ∨ ~ (Y ∧ X)**
0	0	0	1	1
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

Hence, it is a **Tautology**

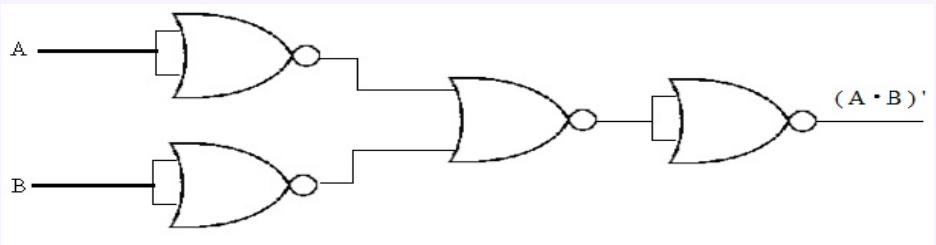
(c) Dual of: Y . X + X' + 1 = 1
= (Y+X) . X' . 0 = 0

(d)

A	**B**	**C**	**D**	**MAXTERMS**	**MINTERMS**
0	1	1	0	A+B'+C'+D	A'.B.C.D'

(e) Logic Circuit of a **NAND** gate using **NOR** gates:

NAND Expression = A'+B' OR (A.B)'



Question 2

(a) Define the term *fall through* condition with reference to switch( ) case. **[2]**

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

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

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

(d) From the class declaration given below, state the nature of the identifiers A, B, C and D: **[2]**

class A extends B implements C, D

(e) State *one* advantage and *one* disadvantage of using *recursion* over *iteration*. **[2]**

Comments of Examiners

(a) This part was well answered by most of the candidates. Some wrote vague definition of 'fall through'. Others explained using examples.

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

(c) Some candidates wrote the answer directly without showing the working. A few candidates made mistakes in calculation while others wrongly determined Row Major instead of Column Major.

(d) Almost all the candidates answered this part correctly. A few candidates were not able to identify the nature of C and D as interfaces.

(e) Various answers were given by the candidates for this part of the question. A few candidates explained with the help of examples. Some wrote definitions of both, without mentioning the advantages and disadvantages, as required.

Suggestions for teachers

- Complexity of a code or a segment must be given more practice. All definition related to complexity must be covered.
- Examples need to be practiced with conversion of Infix to Postfix notation, the order of precedence; the Polish method must be taught.
- More practice should be given in calculating addresses using both Row major and Column major wise. The different terms used in address calculation must be explained to students.
- Java keywords must be explained and practiced in programs for better understanding, especially in inheritance
- The difference between recursion and iteration must be explained; more practice must be given in solving programs by both methods. This will enable students to understand the concept of recursion as compared to iteration.

MARKING SCHEME **Question 2.**	
(a)	**Fall Through** : If a case does not end with a break statement, it will execute the following cases until and unless it counters a break statement. This condition is called *fall through* condition.
(b)	A + B / C * (D / E * F) = A + B / C * (D E / F *) = A + B C / * (D E / F *) = A + B C / D E / F * * **A B C / D E / F * * +**
(c)	**Column Major Wise**: $A[i][j] = BA + W [(j - l_c) * rows + (i - l_r)]$ Putting the values: $1608 = 1500 + 4[(5-1)*rows + (4-1)]$ $1608 = 1500 + 16 *rows + 12$ $-16 * rows = - 96$ **Row = 6**
(d)	A = DERIVED CLASS / SUB CLASS B = SUPER CLASS / BASE CLASS C = INTERFACE D = INTERFACE
(e)	**Advantage**: Recursion makes the complex programs simpler and easy to understand by making the code reduced than Iteration. **Disadvantage**: Recursion occupies more memory and a slow process than iteration.

Question 3

The following function **Check( )** is a part of some class. What will the function **Check( )** return when the values of both 'm' and 'n' are equal to 5? Show the dry run / working. **[5]**

```
int Check (int m, int n)
{
        if (n = = 1)
            return  – m – –;
        else
            return  + + m + Check (m, – –n);
 }
```

Comments of Examiners

A number of candidates were able to answer this question satisfactorily. Common errors made by candidates were:

- Lack of clarity regarding the concept of recursion;
- Confusion with the postfix decrement operator in the return statement of the base case (-m--) and calculating the output as 22 instead of 21.
- Showing the intermediate outputs in memory blocks, without stating the final output.
- Mentioning the correct output without showing the dry-run as required.

Suggestions for teachers

– More practice should be given in solving programs using recursive techniques.

– Recursion and its techniques, must be taught to students with examples.

– Knowledge of base case and recursive case should be given to the students for every program using recursive technique.

– Students should be taught to do systematic dry run in tabular manner. More practice should be given in the concepts of infix and postfix operators.

MARKING SCHEME

Question 3.

OUTPUT : Check(5,5)
6 + Check (6,4)
7 + Check(7,3)
8 + Check(8,2)
9 + Check (9,1)
-9

Hence the output = -9 + 9 + 8 + 7 + 6 = 21

PART – II

*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 (1,3,5,7,8,9,10,11,14,15).**

(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]**

(b) Given the Boolean function:

F(A, B, C, D) = π(4,6,7,10,11,12,14,15).

(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) Most candidates fared well in this part. Some candidates were not able to draw the K-Map for the SOP expression correctly. For a number of candidates the "Map rolling" concept was not very clear. Some converted the canonical form to cardinal form and then reduced it.

(ii) Most of the candidates attempted this part correctly. Some drew the logic circuit using NOR gates while some others drew vague diagrams.

(b)(i) Most candidates answered this question correctly. Some candidates made errors in place value and putting variables in 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.

(ii) Several candidates answered this part correctly. Some drew the logic circuit using NAND gates while some others drew vague diagrams.

Suggestions for teachers

- Make students reduce POS and SOP expressions using K-Map simultaneously. Students should be told not to include the redundant group in the final expression.
- More and 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.
- More and more practice should be given in drawing logic circuits using basic gates and also with universal gates.

MARKING SCHEME

Question 4.

(a) (i) $F(A,B,C,D) = \sum$ (**1 , 3 , 5 , 7 , 8 , 9 , 10 , 11 , 14 , 15**)

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

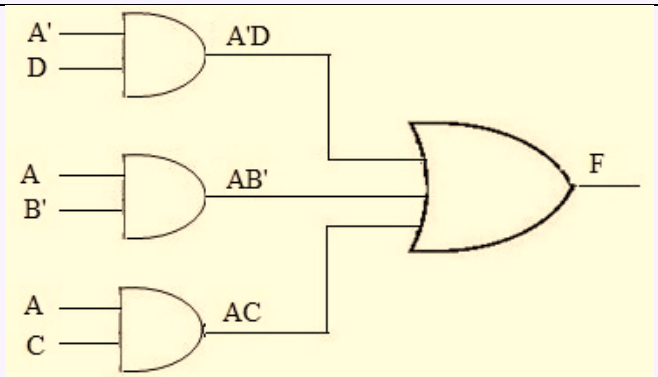
There are three quads :

Quad 1 ($m_1 + m_3 + m_5 + m_7$) = **A'D** Quad2 ($m_8 + m_9 + m_{10} + m_{11}$) = **AB'**

Quad 3 ($m_{10} + m_{11} + m_{14} + m_{15}$) = **AC**

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

(ii)



(b) (i) F(A,B,C,D) = π **(4 , 6 , 7 , 10 , 11 , 12 , 14 , 15)**

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

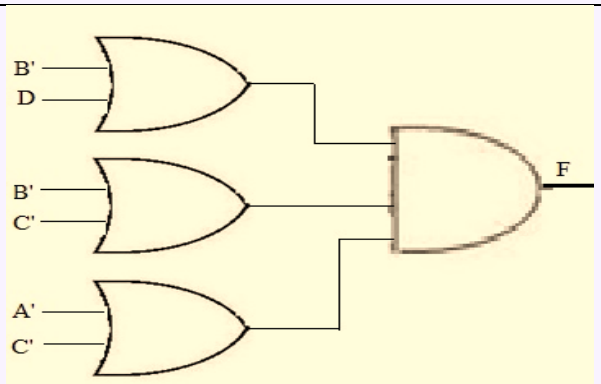
There are three quads :

Quad 1 : ($M_4 M_6 M_{12} M_{14}$) = B' + D Quad 2 : ($M_6 M_7 M_{14} M_{15}$) = B' + C'

Quad 3 : ($M_{10} M_{11} M_{14} M_{15}$) = A' + C'

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

(ii)



Question 5

(a) What is a *decoder?* Draw the logic diagram for a binary to octal (3 to 8) decoder. **[3]**

(b) How is a *half adder* different from a *full adder*? Draw the truth table and derive the SUM and CARRY expression for a full adder. Also, draw the logic diagram for a full adder. **[4]**

(c) State whether the following expression is a Tautology, Contradiction or a Contingency, with the help of a truth table: **[3]**

(X=>Z) V ~ [(X=>Y) Λ (Y=>Z)]

Comments of Examiners

(a) The definition was written correctly by most of the candidates. Some candidates were confused with the logic diagram of a decoder and made the diagram of an encoder instead. A few used the OR gate instead of the AND gate in the logic diagram.

(b) Candidates were able to answer this part correctly. A few candidates did not write the expression and some others drew the logic diagram. The truth table was not proper in some cases.

(c) Most of the candidates answered this part well. Some did not mention whether it was a Tautology, Contradiction or a Contingency. Some were not clear with the symbols 'V' and '∧'.

Suggestions for teachers

- Most practice should be given in drawing encoder and decoder. Use of proper connector and gates must be explained.
- Application of logic gates, like half adders, full adders, encoders, decoders, multiplexers must be explained given the details of the expression, truth table and their logic diagrams.
- More practice must be given in various symbols relating to proposition. Students must be told to read the question properly and answer accordingly.

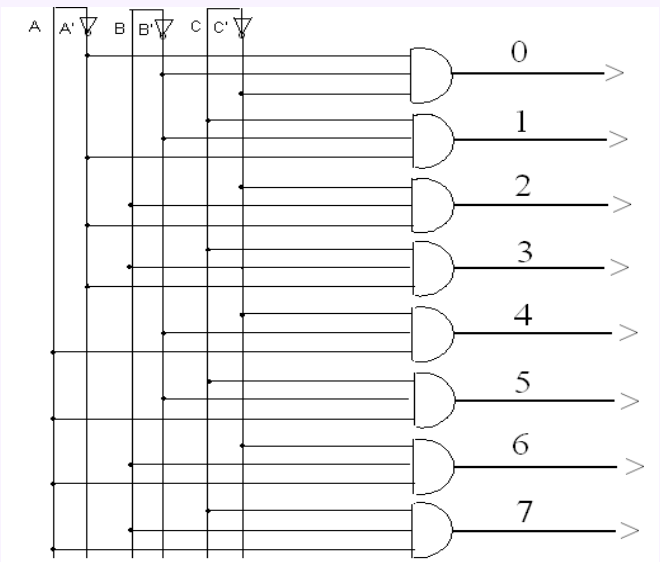
MARKING SCHEME

Question 5.

(a) Decoder: It is a combinational circuit which inputs 'n' lines and outputs 2^n or fewer lines.

It converts LLL to HLL.

Logic Diagram:



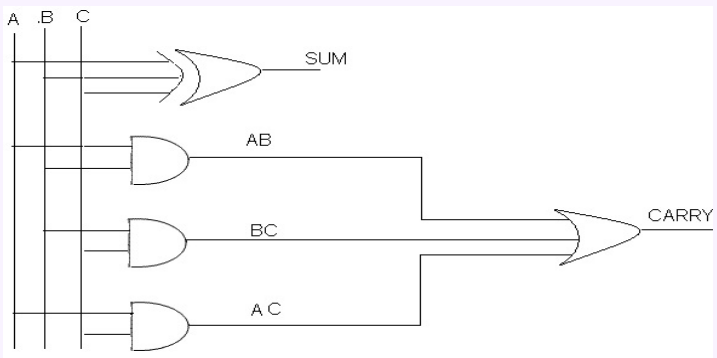
(b) Half adders adds only two binary bits where as a Full adder adds three input binary bits.

Half adder performs partial addition where as a full adder performs total addition.

Truth table for a Full Adder :

A	B	C	SUM(S)	CARRY ZERO (C_0)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Expression for **SUM = A ⊕ B ⊕ C and CARRY = AB + AC + BC**



(c) **Expression**: **(X=>Z) ∨ ~[(X=>Y) ∧ (Y=>Z)]**

X	Y	Z	A	B	C	D	D'	A+D'
			X=>Z	X=>Y	Y=>Z	(X=>Y) ∧ (Y=>Z)		
0	0	0	1	1	1	1	0	1
0	0	1	1	1	1	1	0	1
0	1	0	1	1	0	0	1	1
0	1	1	1	1	1	1	0	1
1	0	0	0	0	1	0	1	1
1	0	1	1	0	1	0	1	1
1	1	0	0	1	0	0	1	1
1	1	1	1	1	1	1	0	1

Hence, it is a **Tautology**.

Question 6

(a) A passenger is allotted a window seat in an aircraft, if he/she satisfies the criteria given below: **[5]**

- The passenger is below 15 years and is accompanied by an adult.

OR

- The passenger is a lady and is not accompanied by an adult.

OR

- The passenger is not below 15 years, but is travelling for the first time.

The inputs are:

INPUTS	
A	The passenger is below 15 years age.
C	The passenger is accompanied by an adult.
L	The passenger is a lady.
F	The passenger is travelling for the first time.

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

Output : **W** – Denotes the passenger is allotted a window seat (1 indicates yes and 0 indicates no)

Draw the truth table for the inputs and outputs given above and write the **SOP** expression for **W(A,C,L,F).**

(b) State the complement properties. Find the complement of the following Boolean expression using De Morgan's law: **[3]**

$AB' + A' + BC$

(c) Differentiate between *Canonical form* and *Cardinal form* of expression. **[2]**

Comments of Examiners

(a) Most candidates attempted this part well. Some did not mention the final expression. A few candidates were confused with the SOP expression and took the output with 0's instead of 1's.

(b) Various laws were mentioned that work on complements. The word 'properties' was not clear to some candidates. Most candidates answered the complementation of the expression correctly.

(c) This part was well answered by most of the candidates. Some represented the difference through examples.

Suggestions for teachers

- Ask students to read the question carefully and answer accordingly so that no part should be left unanswered. More practice should be given in deriving SOP and POS expression from any given truth table (i.e., Minterms and Maxterms).
- The various laws, theorems and properties must be explained separately and their difference explained. More practice should be given in proving laws and properties.
- Different types of expressions and their differences must be explained both with definition and with examples.

MARKING SCHEME

Question 6.

(a)

A	C	L	F	W (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	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	**1**
1	1	1	1	**1**

SOP Expression: W (A ,C, L, F) = $\sum$ (1, 2, 3, 5, 7, 10, 11, 12, 13, 14, 15)

OR

W = A'C'L'F + A'C'LF' + A'C'LF + A'CL'F + A'CLF + AC'LF' + AC'LF +

	ACL'F' + ACL'F + ACLF' + ACLF
(b)	Complement Properties : A + A' = 1 and A . A' = 0 Complement of: AB' + A' + BC = (AB' + A' + BC)' = (AB')' . (A')' . (BC)' = (A'+B) . A . (B' + C')
(c)	Canonical form: If an expression is represented in its maxterms or minterms. Cardinal form: If an expression is represented in its decimal equivalent of its terms.

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 7

A disarium number is a number in which the sum of the digits to the power of their respective position is equal to the number itself. **[10]**

Example: $135 = 1^1 + 3^2 + 5^3$

Hence, 135 is a disarium number.

Design a class **Disarium** to check if a given number is a disarium number or not. Some of the members of the class are given below:

Class name	:	**Disarium**
Data members/instance variables:		
int num	:	stores the number
int size	:	stores the size of the number
Methods/Member functions:		
Disarium(int nn)	:	parameterized constructor to initialize the data members n = nn and size = 0
void countDigit( )	:	counts the total number of digits and assigns it to size
int sumofDigits(int n, int p)	:	returns the sum of the digits of the number(n)

		to the power of their respective positions(p) using **recursive technique**
void check( )	:	checks whether the number is a disarium number and displays the result with an appropriate message

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

Comments of Examiners

Most candidates answered this question correctly. Common errors made by the candidates were:

- The concept of recursion was not clear to some candidates.
- Some did not use the parameters in the function.
- A few candidates wrote the function sum of Digits ( ) without using the recursive technique.
- Several candidates had problem in decrementing the power parameter in the recursive function. The other function including the constructor was well answered. Some candidates did not write the main() function.

Suggestions for teachers

- More practice should be given in solving programs using recursive techniques.
- More attention should be paid by teachers to recursion and its techniques with examples.
- Knowledge of base case and recursive case should be given to students for every program using recursive technique.
- Invoking function within another function should be given more practice.

MARKING SCHEME

Question 7.

```
import java.util.Scanner;
public class Disarium
{  int num,size;
   static Scanner sc=new Scanner(System.in);
   Disarium(int nn)
   {  num=nn;
      size=0;        }
   void countDigits()
   { int a=num;
     while(a!=0)
     { a=a/10;
       size++;         }
   }
   int sumofDigits(int n, int p)
   {  return (n==0)? 0: sumofDigits(n/10,p-1) + (int)Math.pow(n%10,p);    }
   void check()
```

```
  {  if(num==sumofDigits(num,size))
       System.out.print("\n Disarium Number");
     else
       System.out.print("\n Not a Disarium Number");
   }
  static void main()
  {   System.out.println("Input a Number");
      int m=sc.nextInt();
      Disarium x= new Disarium(m);
      x.countDigits();
      x.check();
   }
}
```

Question 8

A class **Shift** contains a two dimensional integer array of order (m×n) where the maximum values of both m and n is 5. Design the class **Shift** to shuffle the matrix (i.e. the first row becomes the last, the second row becomes the first and so on). The details of the members of the class are given below: **[10]**

Class name	:	**Shift**
Data member/instance variable:		
mat[ ][ ]	:	stores the array element
m	:	integer to store the number of rows
n	:	integer to store the number of columns
Member functions/methods:		
Shift(int mm, int nn)	:	parameterized constructor to initialize the data members m = mm and n = nn
void input( )	:	enters the elements of the array
void cyclic(Shift P)	:	enables the matrix of the object(P) to shift each row upwards in a cyclic manner and store the resultant matrix in the current object
void display( )	:	displays the matrix elements

Specify the class **Shift** giving details of the **constructor( )**, **void input( ), void cyclic(Shift)** and **void display( ).** Define the main( ) function to create an object and call the methods accordingly to enable the task of shifting the array elements.

Comments of Examiners

Most candidates answered this question correctly. The cyclic() function was not done properly by some candidates. Various methods/techniques were used to shift the rows in the matrix. Several candidates did it directly without using the parameterized object. Some had problems with the passing of object to the function. Constructor and the main() method was answered properly by candidates.

Suggestions for teachers

- More practice must be given in passing of objects to a function through parameters.
- Working on double dimensional arrays (matrix) must be explained with various examples.

MARKING SCHEME

Question 8.

```
import java.util.Scanner;
public class Shift
{  static Scanner sc=new Scanner(System.in);
   int mat[][];
   int m,n;
   Shift(int mm,int nn)
   { m=mm;
     n=nn;
     mat=new int[m][n];
   }
   void input()
   {  System.out.println("Enter elements");
     for(int i=0;i<m;i++)
      for(int j=0;j<n;j++)
       mat[i][j]=sc.nextInt();
   }
   void display()
   {
      for(int i=0;i<m;i++)
      {  System.out.println();
        for(int j=0;j<n;j++)
          System.out.print(mat[i][j] +"\t");
      }
   }
   void cyclic(Shift P)
   {  for(int i=0;i<m;i++)
        for(int j =0;j<n;j++)
          {    if(i!=0)
                    mat[i-1][j]=P.mat[i][j];
               else
                    mat[m-1][j]=P.mat[0][j];
           }
    }
```

```
    static void main()
    { Shift x=new Shift(3,4);
      Shift y=new Shift(3,4);
      x.input();
      y.cyclic(x);
      x.display();
      y.display();
    }
}
```

Question 9

A class **ConsChange** has been defined with the following details: **[10]**

Class name	:	**ConsChange**
Data members/instance variables:		
word	:	stores the word
len	:	stores the length of the word
Member functions/methods:		
ConsChange( )	:	default constructor
void readword( )	:	accepts the word in lowercase
void shiftcons( )	:	shifts all the consonants of the word at the beginning followed by the vowels (e.g. spoon becomes spnoo)
void changeword( )	:	changes the case of all occurring consonants of the shifted word to uppercase, for e.g. (spnoo becomes SPNoo)
void show( )	:	displays the original word, shifted word and the changed word

Specify the class **ConsChange** giving the details of the **constructor( ), void readword( )**, **void shiftcons( ), void changeword( )** and **void show( )**. Define the main( ) function to create an object and call the functions accordingly to enable the task.

Comments of Examiners

Most candidates were able to attempt this question well. Different methods / logic were used to shift consonants in the shiftcons() function and to change case in the changeword() function. Some candidates included local variables in the functions and shared it with other functions. Some were not able to display the required output in the show() function. The main() function and constructor were not answered by a few 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 candidates. Knowledge of constructors to initialize a string and other data members should be given.

MARKING SCHEME

Question 9.

```
import java.util.Scanner;
public class ConsChange
{   String word;
    int len;
    static Scanner sc=new Scanner(System.in);
 ConsChange()
 {  len=0; word= “”;  }
 void readword()
 { System.out.println("Enter word in Lower case");
   word=sc.next();
   len=word.length();  }
 void shiftcons()
 { String s="";  char c;
    for(int i=0;i<len;i++)
    { c=word.charAt(i);
      if(c!='a' && c!='e' && c!='i' && c!='o' && c!='u')
          s +=c;
     }
     for(int i=0;i<len;i++)
     { c=word.charAt(i);
          if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
            s +=c;  }
     System.out.print("\n Sorted Word="+s);
     word=s;
   }
 void changeword()
 { char c; String s="";
    for(int i=0;i<len;i++)
      { c=word.charAt(i);
        if(c!='a' && c!='e' && c!='i' && c!='o' && c!='u')
```

```
          s +=(char)(c-32);
        else
          s +=c;
       }
       System.out.println("\n Changed word= " + s);     }
  void show()
  {   System.out.print("\n Original word= " + word);
      shiftcons();
      changeword();       }
   static void main()
   { ConsChange X=new ConsChange();
     X.readword();
     X.show();
   }
}
```

SECTION – C

*Answer **any two** questions.*

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

(Flowcharts are **not** required.)

Question 10

A super class **Bank** has been defined to store the details of a customer. Define a sub-class **Account** that enables transactions for the customer with the bank. The details of both the classes are given below: **[5]**

Class name	:	**Bank**
Data member/instance variable:		
name	:	stores the name of the customer
accno	:	stores the account number
p	:	stores the principal amount in decimals
Member functions/methods:		
Bank(…)	:	parameterized constructor to assign values to the instance variables
void display( )	:	displays the details of the customer
Class name:		**Account**
Data member/instance variable:		
amt	:	stores the transaction amount in decimals
Member functions/methods:		
Account(…)	:	parameterized constructor to assign values to the instance variables of both the classes
void deposit( )	:	accepts the amount and updates the principal as p=p + amt

void withdraw( )	:	accepts the amount and updates the principal as p=p−amt If the withdrawal amount is more than the principal amount, then display the message "INSUFFICIENT BALANCE". If the principal amount after withdrawal is less than 500, then a penalty is imposed by using the formula p=p−(500−p)/10
void display( )	:	displays the details of the customer

*Assume that the super class **Bank** has been defined*. Using the **concept of Inheritance**, specify the class **Account** giving details of the **constructor(…)**, **void deposit( )**, **void withdraw( )** and **void display( ).**

The super class and the main function need not be written.

Comments of Examiners

The 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 several candidates.

A few candidates declared the base class also, which was not required. Data members were not declared properly by some students. The function withdraw() 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 by the candidates.

Suggestions for teachers

- Practice should be given on inheritance. Use of constructor using the base class member must be made clear.
- Explain 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.
- Instruct the students to read the question properly (base class not required) and answer accordingly.

MARKING SCHEME

Question 10.

```
 import java.util.Scanner;
public class Account extends Bank
{  static Scanner sc=new Scanner(System.in);
    double amt;
   Account(String n, String a, double pp)
  {     super(n,a,pp);
         amt=0.0;
   }
  void deposit( )
  {     System.out.print("\n Enter amount");
        amt=sc.nextDouble();
        p=p-amt;
```

```
  }
  void withdraw()
  {    System.out.print("\n Enter amount");
       amt=sc.nextDouble();
       if(amt>p)
           System.out.println("INSUFFICIENT BALANCE");
      else
        {  p=p-amt;
           if(p<500)
             p=p-(500-p)/10;
         }
   }
   void display()
   {    super.display();
    }
}
```

Question 11

A bookshelf is designed to store the books in a stack with LIFO(Last In First Out) operation. Define a class **Book** with the following specifications: **[5]**

Class name	:	**Book**
Data members/instance variables:		
name[ ]	:	stores the names of the books
point	:	stores the index of the topmost book
max	:	stores the maximum capacity of the bookshelf
Methods/Member functions:		
Book(int cap)	:	constructor to initialise the data members max = cap and point = −1
void tell( )	:	displays the name of the book which was last entered in the shelf. If there is no book left in the shelf, displays the message “SHELF EMPTY”
void add(String v)	:	adds the name of the book to the shelf if possible, otherwise displays the message “SHELF FULL”
void display( )	:	displays all the names of the books available in the shelf

Specify the class **Book** giving the details of **ONLY** the functions **void tell( )** and **void add(String)**. Assume that the other functions have been defined.

The main function need not be written.

Comments of Examiners

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

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

(ii) increment / decrement of top index was not done properly. Some candidates used integer as stack type instead of string array. The methods tell() and add() was found to be difficult by some of the candidates. In some cases, the constructor was also defined which was not required. Rest of the program was well answered.

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 real world examples.
- 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 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. So, the array is used to show the working of a stack, queue or a de queue.

MARKING SCHEME

Question 11.

```
 public class Book
{  String name[ ];
   int point,max;
   Book(int cap)
   { max=cap;
     point = -1;
     name=new String[max];
   }
   void tell()
   {  if(point<0)
        System.out.print("\n SHELF EMPTY");
      else
        System.out.print("\n " + name[point]);
   }
   void add(String v)
   { if(point<max-1)
```

```
      name[++point]=v;
    else
      System.out.print("\n SHELF FULL");
   }
  void display()
  { for(int i=point;i>=0;i--)
      System.out.print("\n " + name[i]);
  }
}
```

Question 12

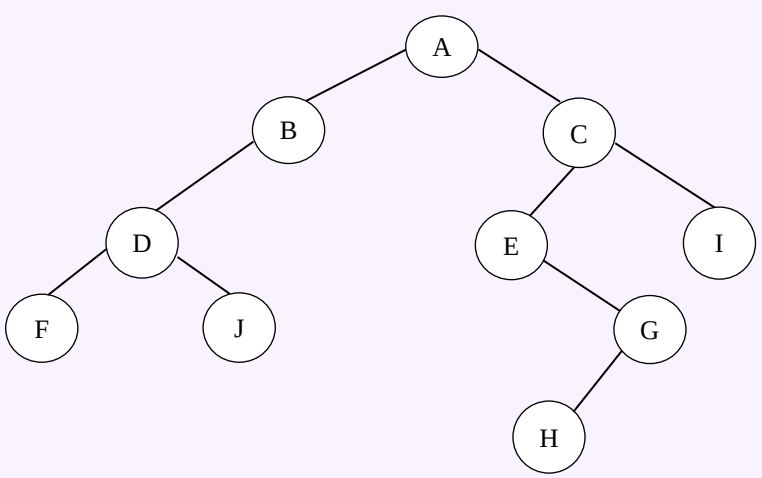
(a) A linked list is formed from the objects of the class Node. The class structure of the Node is given below: **[2]**

```
class Node
{
        String name;
        Node   next;
}
```

Write an *Algorithm* **OR** a *Method* to search for a given name in the linked list. The method of declaration is given below:

boolean searchName(Node start, String v)

(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) Name the parent of the nodes B and G. **[1]**

(iii) Name the leaves of the right sub-tree. **[1]**

Comments of Examiners

(a) Most candidates answered this part correctly and scored full credit. Some candidates had problems in moving the pointer to the next node and checking for null. Some wrote the algorithm in simple English covering all the main steps.

(b) (i) This part was answered correctly by most of the candidates but some wrote the post order and the pre order of the tree.

(ii) Almost all candidates answered this part correctly.

(iii) Most of the candidates answered this part correctly. A few candidates mentioned the leaves of the tree instead of the right sub tree.

Suggestions for teachers

- More programs / algorithms should be practiced with link list and binary tree data structure. Use of diagrams to illustrate the link list and the Binary Trees must be practiced.
- 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)

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
Step 4. Search for the node. If found display details, exit
Step 5. Move pointer to the next node
Step 6. End

METHOD:

```
boolean searchName(Node start, String v)
 {  Node temp=new Node(start);
    while(temp.name.equals(v) = = false && temp.next!=null)
        temp=temp.next;
    if(temp.next!=null)
        return true;
    else
        return false;
 }
```

(b)	(i)	F D J B A E H G C I
	(ii)	Parent of B=A and Parent of G=E
	(iii)	H and I

GENERAL COMMENTS:

(a) Topics found difficult by candidates:

- The symbols ‘∧’ and ‘v’ from propositional logic
- ‘fall through’ condition in switch case()
- Keywords ‘extends’ and ‘implements’ with respect to inheritance
- Returning value of the base case in recursive output
- K-MAPS (Grouping , map-rolling , place value)
- Complement Properties
- Recursive technique
- Passing objects to functions
- Stack operations for adding and removing stings

(b) Concepts in which candidates got confused:

- The symbols in a proposition.
- The terms ‘extends’ and ‘implements’
- Symbol of decrement operator(postfix)
- Passing objects to functions
- Concatenation of two strings
- Cyclic shift in matrix.
- Use of Single instance variable for multiple operations in various functions.

(c) Suggestions for Candidates:

- Prepare summary for each chapter.
- Answers and definitions should be short and precise and according to marks intended.
- Important words and terms should be underlined. 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.
- 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.
- The Scope of the syllabus should be followed.