COMPUTER SCIENCE

PAPER 1

(THEORY)

Three hours

*(Candidates are allowed additional 15 minutes for **only** reading the paper. They must NOT start writing during this time.)*

---

*Answer **all** questions in Part I (compulsory) and **seven** questions from Part-II, choosing **three** questions from Section-A, **two** from Section-B and **two** from Section-C .*

*All working, including **rough** work, should be done on the same sheet as the rest of the answer.*

The intended marks for questions or parts of questions are given in brackets [ ].

PART I

Answer **all** *questions.*

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

Question 1

(a) State the Principle of Duality. Write the dual of: **[2]**

$(P+Q').R.1 = P.R + Q'.R$

(b) Minimize the expression using Boolean laws: **[2]**

$F = (A + B').(B + CD)'$

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

$F(P,Q,R) = \pi(1,3)$

(d) Using a truth table verify: **[2]**

$(\sim p \Rightarrow q) \wedge p = (p \wedge \sim q) \vee (p \wedge q)$

(e) If A = 1 and B = 0, then find: **[2]**

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

(ii) $(A + B')'$

Question 2

(a) Differentiate between *throw* and *throws* with respect to exception handling. [2]

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

$$E* (F / (G - H) * I) + J$$

(c) Write the algorithm for push operation (to add elements) in an array based stack. [2]

(d) Name the File Stream classes to: [2]

(i) Write data to a file in binary form.

(ii) Read data from a file in text form.

(e) A square matrix M [ ][ ] of size 10 is stored in the memory with each element requiring 4 bytes of storage. If the base address at M [0][0] is 1840, determine the address at M [4][8] when the matrix is stored in **Row Major Wise**. [2]

Question 3

(a) The following function **Recur ( )** is a part of some class. What will be the output of the function **Recur** ( ) when the value of n is equal to 10. Show the dry run / working. [5]

```
void Recur (int n)
{
    if (n > 1 )
    {
        System.out.print ( n + " " );
        if (n%2 !=0)
        {
            n = 3 * n+1;
            System.out.print(n + " ");
        }
        Recur (n/2);
    }
}
```

(b) The following function is a part of some class. Assume 'n' is a positive integer. Answer the given questions along with dry run / working.

```
int unknown (int n)
{
    int i, k;
    if (n%2==0)
    {   i = n/2;
        k=1;
    }
    else
    {
        k=n;
        n--;
        i=n/2;
    }
    while (i > 0)
    {
        k=k*i*n;
        i--;
        n--;
    }
    return k;
}
```

(i) What will be returned by unknown(5)? **[2]**

(ii) What will be returned by unknown(6)? **[2]**

(iii) What is being computed by unknown (int n)? **[1]**

PART – II

*Answer **seven** questions in this part, choosing **three** questions from Section A, **two** from Section B and **two** from Section C.*

SECTION - A

*Answer any **three** questions.*

Question 4

(a) Given the Boolean function: **$F(A,B,C,D) = \Sigma(0, 2, 4, 5, 8, 9, 10, 12, 13)$**

(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. Assume that the variables and their complements are available as inputs. **[1]**

(b) Given the Boolean function: $F(P,Q,R,S) = \pi(0, 1, 3, 5, 7, 8, 9, 10, 11, 14, 15)$

(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. Assume that the variables and their complements are available as inputs. **[1]**

Question 5

A Football Association coach analyzes the criteria for a win/draw of his team depending on the following conditions.

- If the Centre and Forward players perform well but Defenders do not perform well.

OR

- If Goal keeper and Defenders perform well but the Centre players do not perform well.

OR

- If all the players perform well.

The inputs are :

INPUTS	
C	Centre players perform well.
D	Defenders perform well.
F	Forward players perform well.
G	Goalkeeper performs well.

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

Output: **X** - Denotes the win/draw criteria [1 indicates win/draw and 0 indicates defeat in all cases.]

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

(b) Reduce **X (C, D, F, G)** using Karnaugh's Map. **[5]**
Draw the logic gate diagram for the reduced **POS** expression for X (C, D, F, G) using AND and OR gate. You may use gates with two or more inputs. Assume that the variable and their complements are available as inputs.

Question 6

(a) In the following truth table x and y are inputs and B and D are outputs: [3]

INPUT		OUTPUT	
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Answer the following questions:

(i) Write the SOP expression for D.

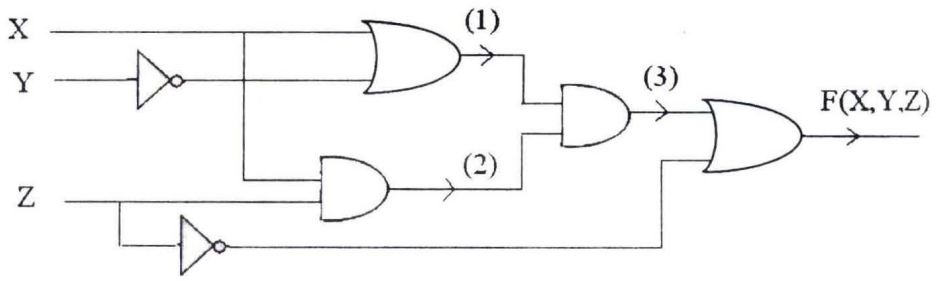
(ii) Write the POS expression for B.

(iii) Draw a logic diagram for the SOP expression derived for D, using only NAND gates.

(b) Using a truth table, verify if the following proposition is valid or invalid: [3]

$$(a => b) \wedge (b => c) = (a => c)$$

(c) From the logic circuit diagram given below, name the outputs (1), (2) and (3). Finally derive the Boolean expression and simplify it to show that it represents a logic gate. Name and draw the logic gate. [4]



Question 7

(a) What are Decoders? How are they different from Encoders? [2]

(b) Draw the truth table and a logic gate diagram for a 2 to 4 Decoder and briefly explain its working. [4]

(c) A combinational logic circuit with three inputs P, Q, R produces output 1 if and only if an odd number of 0's are inputs. **[4]**

(i) Draw its truth table.

(ii) Derive a canonical SOP expression for the above truth table.

(iii) Find the complement of the above derived expression using De Morgan's theorem and verify if it is equivalent to its POS expression.

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

An emirp number is a number which is prime backwards and forwards. Example: 13 and 31 are both prime numbers. Thus, 13 is an emirp number. **[10]**

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

Class name : **Emirp**

Data members/instance variables:

n	:	stores the number
rev	:	stores the reverse of the number
f	:	stores the divisor

Member functions:

Emirp(int nn)	:	to assign n = nn, rev = 0 and f = 2
int isprime(int x)	:	check if the number is prime using the **recursive technique** and return 1 if prime otherwise return 0
void isEmirp( )	:	reverse the given number and check if both the original number and the reverse number are prime, by invoking the function isprime(int) and display the result with an appropriate message

Specify the class **Emirp** giving details of the **constructor(int), int isprime(int)** and **void isEmirp( ).** Define the **main( )** function to create an object and call the methods to check for Emirp number.

Question 9

Design a class **Exchange** to accept a sentence and interchange the first alphabet with the last alphabet for each word in the sentence, with single letter word remaining unchanged. The words in the input sentence are separated by a single blank space and terminated by a full stop. **[10]**

Example: Input: It is a warm day.

Output: tI si a marw yad

Some of the data members and member functions are given below:

Class name	:	**Exchange**

Data members/instance variables:

sent	:	stores the sentence
rev	:	to store the new sentence
size	:	stores the length of the sentence

Member functions:

Exchange( )	:	default constructor
void readsentence( )	:	to accept the sentence
void exfirstlast( )	:	extract each word and interchange the first and last alphabet of the word and form a new sentence rev using the changed words
void display( )	:	display the original sentence along with the new changed sentence.

Specify the class **Exchange** giving details of the **constructor( ), void readsentence( ), void exfirstlast( )** and **void display( )**. Define the **main( )** function to create an object and call the functions accordingly to enable the task.

Question 10

A class **Matrix** contains a two dimensional integer array of order [m × n]. The maximum value possible for both 'm' and 'n' is 25. Design a class **Matrix** to find the difference of the two matrices. The details of the members of the class are given below: **[10]**

Class name	:	**Matrix**
Data members/instance variables:		
arr[ ][ ]	:	stores the matrix element
m	:	integer to store the number of rows
n	:	integer to store the number of columns
Member functions:		
Matrix (int mm, int nn)	:	to initialize the size of the matrix m = mm and n = nn
void fillarray( )	:	to enter the elements of the matrix
Matrix SubMat(Matrix A)	:	subtract the current object from the matrix of parameterized object and return the resulting object
void display( )	:	display the matrix elements

Specify the class **Matrix** giving details of the **constructor(int,int), void fillarray( ), Matrix SubMat(Matrix)** and **void display**. Define the **main( )** function to create objects and call the methods accordingly to enable the task.

SECTION – C

*Answer any **two** questions.*

Each program should be written in such a way that it clearly depicts the logic of the problem stepwise.

This can be achieved by using comments in the program and mnemonic names or pseudo codes for algorithms. The programs must be written in Java and the algorithms must be written in general / standard form, wherever required / specified.

(Flowcharts are **not** required.)

Question 11

A super class **Perimeter** has been defined to calculate the perimeter of a parallelogram. Define a subclass **Area** to compute the area of the parallelogram by using the required data members of the super class. The details are given below: **[10]**

Class name	:	**Perimeter**
Data members/instance variables:		
a	:	to store the length in decimal
b	:	to store the breadth in decimal
Member functions:		
Perimeter(...)	:	parameterized constructor to assign values to data members
double Calculate( )	:	calculate and return the perimeter of a parallelogram as 2 * (length + breadth)
void show( )	:	to display the data members along with the perimeter of the parallelogram
Class name	:	**Area**
Data members/instance variables:		
h	:	to store the height in decimal
area	:	to store the area of the parallelogram
Member functions:		
Area(...)	:	parameterized constructor to assign values to data members of both the classes
void doarea( )	:	compute the area as (breadth * height)
void show( )	:	display the data members of both classes along with the area and perimeter of the parallelogram.

Specify the class **Perimeter** giving details of the **constructor(...), double Calculate( )** and **void show( ).** Using the **concept of inheritance**, specify the class **Area** giving details of the **constructor(...), void doarea( )** and **void show( ).**

The main function and algorithm need not be written.

Question 12

A doubly queue is a linear data structure which enables the user to add and remove integers from either ends, i.e. from front or rear. Define a class **Dequeue** with the following details: **[10]**

Class name	:	**Dequeue**
Data members/instance variables:		
arr[ ]	:	array to hold up to 100 integer elements
lim	:	stores the limit of the dequeue
front	:	to point to the index of front end
rear	:	to point to the index of the rear end
Member functions:		
Dequeue(int l)	:	constructor to initialize the data members lim=l; front=rear=0
void addfront(int val)	:	to add integer from the front if possible else display the message ("Overflow from front")
void addrear(int val)	:	to add integer from the rear if possible else display the message ("Overflow from rear")
int popfront( )	:	returns element from front, if possible otherwise returns - 9999
int poprear( )	:	returns element from rear, if possible otherwise returns - 9999

Specify the class **Dequeue** giving details of the **constructor(int), void addfront(int), void addrear(int), int popfront( )** and **int poprear( ).**

The main function and algorithm need not be written.

Question 13

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

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

Write an *Algorithm* **OR** a *Method* to count the number of nodes in the linked list. The method declaration is given below:

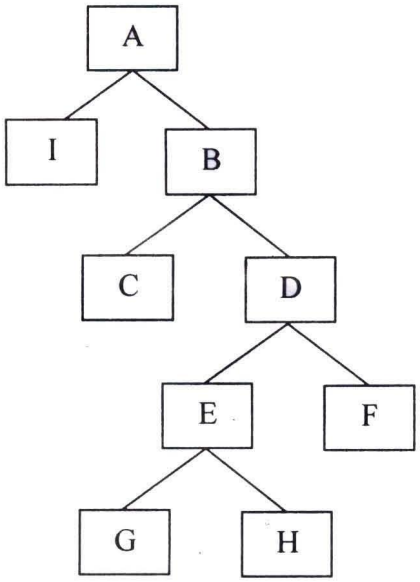
int count(Node ptr_start)

(b) What is the Worst Case complexity of the following code segment: **[2]**

(i)
```
for (int p = 0; p < N; p++)
  {
   for (int q=0; q < M; q++)
        {
          Sequence of statements;
        }
  }
 for (int r = 0; r < X; r++)
    {
         Sequence of statements;
    }
```

(ii) How would the complexity change if all the loops went upto the same limit N?

(c) Answer the following from the diagram of a Binary Tree given below: **[4]**



(i) Preorder Transversal of tree.

(ii) Children of node E.

(iii) Left subtree of node D.

(iv) Height of the tree when the root of the tree is at level 0.