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) Verify using the truth table, if (X=>Y)**.**(Y=>X) is a *tautology*, *contradiction* or *contingency*. **[2]**

(b) State the two Idempotence laws of Boolean Algebra. Verify any one of them using Truth Table. **[2]**

(c) Show how a NAND gate can be used to construct an OR gate. **[2]**

(d) Given, F (X, Y, Z) = (X+Y)**.**(Y+Z) write the function in canonical product-of-sum form. **[2]**

(e) Given, the Boolean Function, F (X, Y, Z) = ∑ (2, 3, 4, 6, 7). **[2]**
Reduce it using Karnaugh's Map. And also find the complement of its result.

Question 2

(a) State the difference between Function Overloading and Function Overriding. **[2]**

(b) What is a Binary Tree? What do you mean by Traversing a Tree? **[2]**

(c) Each element of an array A [-15….. 20, 20….. 45] requires 4 bytes for storage. **[2]**
Find the address of A [30][20] if the base address is 1000, in:
(i) Row Major Wise
(ii) Column Major Wise

(d) What do you mean by Complexity? What is the complexity of Binary Search? **[2]**

(e) Convert the following infix notation to its postfix form: **[2]**
(A + (B * C)) / (C – (D * B))

Question 3

(a) Give output of the following function where x and y are arguments greater than 0. Show the dry run/working.

```
void confusing ( int x, int y )
{
  if ( x > 1 ) // base case
   {
    if ( x % y = = 0 )
      {
        System.out.print( y + " " );
        confusing ( x / y, y);
       } // end of inner if
    else
        confusing ( x , y+1);
   } //end of outer if
}
```

(i) What will the function confusing (24 , 2) return ? **[2]**
(ii) What will the function confusing (84 , 2) return ? **[2]**
(iii) In one line, state what the function is doing, apart from recursion. **[1]**

(b) The following function is a part of some class which prints whether a number is a **Magic Number** or not. It returns the value 1 when the number is a Magic Number, otherwise it returns 0.
There are some places in the code marked by ?1?, ?2?, ?3?, ?4?, ?5? which must be replaced by a statement / expression so that the function works properly:

/* A Magic Number is a number whose sum of the digits equals to 1, when this addition of digits is performed till the number itself becomes a single digit number. **Example:** 289, adding 2+8+9 it gives 19, then add, 1+9 it gives 10, then add 1+0, it gives 1, which is a single digit number, and since its sum is equal to 1, so, 289 is a magic number */

```
int isMagic (int n)
{
   int dig = 0, s = n;
   while ( ? 1 ? )
    {
      n = s, s = 0;
      while (n > 0)    {
         dig = ? 2 ?;
         s = s + ? 3 ?;
         ? 4 ?;          } //end of inner while
    } //end of outer while
if ( ? 5 ? )
    return 1;
else
    return 0;
}
```

(i) What is the expression/value at **? 1 ?** **[1]**
(ii) What is the expression/value at **? 2 ?** **[1]**
(iii) What is the expression/value at **? 3 ?** **[1]**
(iv) What is the expression/value at **? 4 ?** **[1]**
(v) What is the expression/value at **? 5 ?** **[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) = Σ (0, 1, 2, 5, 8, 9, 10)**

(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 variable and their complements are available as inputs. **[1]**

(b) Given the Boolean function: **F (A,B,C,D) = π (3, 4, 6, 7, 11, 12, 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. Assume that the variable and their complements are available as inputs. **[1]**

Question 5

The past pupil Association of R. K. University Computer Science Department is organizing a reunion function at the campus. The invitation card is to be issued to a person if:

- The person is an ex-student of the department and had passed out in 1995.

or,

- The person is not an ex-student of the same department but passed out from the University in 1995 and has made a contribution of ` 1000.

THE INPUTS ARE:

E : The person is an ex-student of the department
U : The person is not an ex-student of the department but a student of the same university
P : The person passed out in
C : The person contributes ₹ 1000

OUTPUT IS:

I : Denotes the Invitation Card is issued[1 indicates it is issued and 0 indicates it is not issued]

(a) Draw the truth table for the inputs and outputs given above and write the **SOP** expression for I (E, U, P, C). **[5]**

(b) Reduce I (E, U, P, C) using Karnaugh's map. **[5]**

Draw the logic gate diagram for the reduced SOP expression for I(M, S, C, Y) using AND & OR gates.
You may use gates with two or more inputs. Assume that variable and their complements are available as inputs.

Question 6

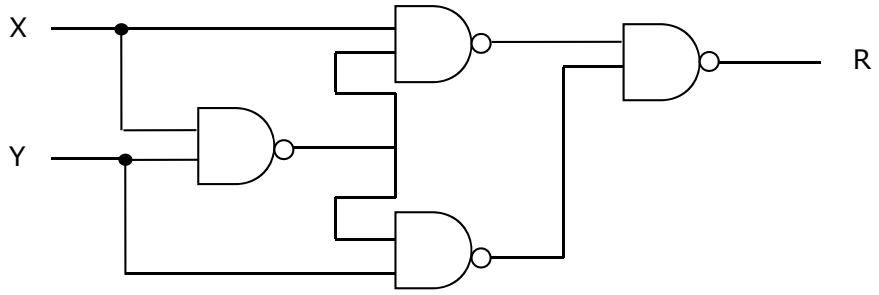
(a) Draw the logic diagram and Truth Table to Encode the Hexadecimal lines (A – F). Briefly explain the working of the logic diagram. **[5]**

(b) Prove that the complement of A**.**(A + B)**.**B**.**(B + $\bar{C}$) is a Universal Gate. **[3]**

(c) Given : F (x, y, z) = Π (2,3,6,7) **[2]**
Verify : F (x, y, z) = Σ (0,1,4,5)

Question 7

(a) Find out the Boolean expression R from the following Logic diagram : **[3]**



Can you replace this by a single gate? If yes, draw that logic gate.

(b) State any one of the applications the following: **[2]**

(i) Multiplexer
(ii) Half Adder

(c) Using only NAND gates, draw a Logic circuit of the following function: **[2]**
F (X, Y, Z) = X**.**Y' + X'**.**Z

(d) What are decoders? **[2]**

(e) How many select lines does an **8 : 1** multiplexer have? **[1]**

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 Combination function C(n , k) gives the number of different (unordered) K – elements Subsets that can be found in a given set of n elements.

The function can be computed from the formula:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Design a class Combination to implement this formula. Some of the data members and member functions are given below.

Class name	**:**	**Combination**
Data members/instance variables	**:**	
n	:	integer number
k	:	integer number
Member functions :		
Combination ( )	:	to initialize the data members n = 0 and k = 0
void read( )	:	to accept the value of the data members
int fact(int)	:	return the factorial of a number using **Recursion Technique**.
void compute( )	:	calculate and display the result

(a) Specify the class **Combination**, giving details of the **Constructor** and member functions **void read( ), int fact(int) and void compute( )**. Also write the **main( )** function to create an object and call the member function accordingly in order to enable the task . **[8]**

(b) Give any two appropriate differences between ***recursion*** and ***iteration*** process. **[2]**

Question 9 **[10]**

A class MyArray contains an array of n integers (n<=100) that are already arranged in ascending order. The subscripts of the array elements vary from 0 to n-1. Some of the member functions of the class are given below:

Class name : **MyArray**

Data members/instance variables :

arr	:	an array of n integers.
n	:	size of the array

Member functions/methods :

MyArray( )	:	constructor to initialize 0 in each memory location of the array arr[ ]
void readArray( )	:	to read n integers that are arranged in ascending order.
void displayArray( )	:	to display the n integers of the array arr [ ]
int binarySearch(int)	:	to search for the value taken as parameter in the array using the Binary Search Technique. It returns the subscript of the array element if the value is found, otherwise it returns -999.

(a) Specify the class MyArray giving the details of the **void displayArray( )** and **int binarySearch(int)**. You **need not** write the main function. **[7]**

(b) What changes would be necessary in the method **binarySearch(int)** if the given array was arranged in descending order. **[2]**

(c) When will binary search technique fail to work? **[1]**

Question 10 **[10]**

Design a class **Palindrome** which enables a word to be arranged in ascending order according to its alphabets and also finds the frequency of each alphabets present in the word. The details of the members of the class are given below:

Class name : **Palindrome**

Data members/instance variables :

str	:	to store a word
revword	:	to store the reverse of the word

Member functions/methods :

Palindrome( )	:	constructor to initialize data members with initial values
void readWord( )	:	to accept the inputted word
String rev(String, int)	:	to reverse the word using **Recursive Technique**
void disp( )	:	to display the original word, and the reversed word
void isPalin( )	:	to check whether the inputted word is a Palindrome or not by calling the **rev( )** function.

Specify the class **Palindrome** giving details of the **constructor** and member functions **void readWord( ), String rev(String,int), void disp( )** and **void isPalin( )**. Define the **main( )** function to create an object and call the member function according to enable the task .

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, wherever required.

Question 11 **[10]**

A library issues a book on rental basis at a 2% charge on the cost price of the book per day. As per the library, a book can be retained for 7 days without any fine. If the book is returned after 7 days, a fine will also be charged for the excess days as per the chart given below :

Number of excess days	Fine per day (₹)
1 to 5	2.00
6 to 10	3.00
Above 10 days	5.00

Design a class Library and another class Compute to perform the task. The classes details are as given:

Class name : **Library**

Data members/instance variables :

Name, author : string variables to store name of book and authors name
p : Price of the book in decimals.

Member functions/methods :

Library(String n,String a,double np) : parameterized constructor to assign the parameters n to name, a to author and np to p.

void show( ) : displays the book details

Class Name : **Compute**

Data members/instance variables :

d : number of days taken in returning the book
r : to store the fine.

Member functions/methods :

Compute (.....) : parameterized constructor to assign values to data members of **both** the classes.
void fine( ) : calculates the fine for the excess days.
void display( ) : displays the book details along with the number of days, fine and total amount to be paid as fine. The Amount is calculated as :
(2% of price of book * total no. of days) + fine

Specify the class **Library** giving the details of the **constructor** and **void show( )**. Using the concept of **inheritance**, specify the class **Compute** giving the details of constructor, **void Fine( )** and **void display( )** function. You **do not** need to write the main( ) function.

Question 12

Rack is a kind of data structure which can store at the most 20 books. The rack restriction is that a book can be kept in to the rack or removed only at one end i.e. on the top.

The class Rack has the following details :

Class name : Rack

Data members/ instance variables :

book[ ]	: array of string of maximum 50 locations to store books.
name	: string variable to store name of book.
Limit	: integers as maximum capacity of the array
top	: integer to indicate topmost book into the Rack.

Member functions/methods :

Rack( )	: a constructor to store blank in the array book[ ].
Rack(int nx)	: a constructor to assign ***nx*** to **limit** and -1 to top.
void dispList( )	: to display list of books in the rack.
boolean ifEmpty( )	: returns true, if rack is empty, otherwise returns false.
void putTheBook( )	: input name of the book into variable name and adds it on the top into the array book[ ] if rack is empty otherwise prints a message "Rack overflow".
void removeBook( )	: removes a book form the top of the rack, if rack is not Empty and print the book name, otherwise outputs a Message "Rack underflow".

(a) Specify the class **Rack** giving details of **constructors** and the functions **ifEmpty()**, **void putTheBook( )**, **void removeBook( )** only. Assume that other functions are already written for you. You **do not need** to write the main function. **[9]**

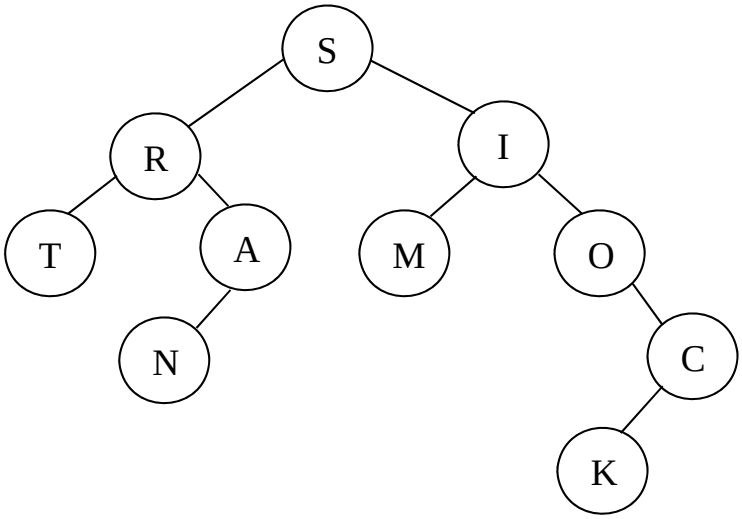
(b) Name the entity which is being used by the above class and also state the principle it uses. **[1]**

Question 13

(a) Link Lists are linear data structure. Create Algorithms for the following operations: **[4]**

(i) Insertion of a Node at the beginning in a Link List.
(ii) Deletion of the first Node in a Link list.

(b) Answer the questions below for the given binary tree: **[4]**



(i) State the Height of the Binary Tree
(ii) List the leaf nodes of the Binary Tree
(iii) Write In-order traversal of the Binary Tree
(iv) Write Preorder traversal of the Binary Tree

(c) State the **Best Case** and **Worst Case** complexity for the following algorithms : **[2]**

(i) Bubble Sort
(ii) Selection Sort