COMPUTER SCIENCE

Paper – 1

(THEORY)

(Three hours)

Maximum Marks: 70

*(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 **six** questions from Part-II, choosing **two** 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 Associative law and prove it with the help of a truth table. **[1]**

(b) Draw the truth table to prove the proportional logic expression. **[1]**

$$(x => y) \wedge (y => x) = x <=> y$$

(c) Find the dual for the Boolean equation: AB' + BC' + 1 = 1. **[1]**

(d) Convert the Boolean expression F(X,Y,Z) = X'Y'Z + X'YZ' + XYZ into its cardinal form. **[1]**

(e) Minimize F = XY + (XZ)' +XY'Z using Boolean laws. **[1]**

Question 2

(a) Differentiate between *Stack data structure* and *Queue data structure.* **[2]**

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

A * (B / C) / E + F

(c) Define *Interface*. How is it different from a Class? **[2]**

(d) Each element of an array arr[15][20] requires 'W' bytes of storage. If the address of arr[6][8] is 4440 and the Base Address at arr[1][1] is 4000 , find the width '**W**' of each cell in the array arr[ ][ ] when the array is stored as **Column Major Wise.** **[2]**

(e) Define Big 'O' notation. State the two factors which determine the complexity of an algorithm. **[2]**

Question 3

The following is a function of some class. What will be the output of the function **test ( )** when the value of count is equal to 4 ? Show the dry run / working. **[5]**

```
void test (int count )
{
        if ( count = = 0)
            System.out.println(" " );
        else
        {  System.out.println( "Bye" + count);
            test( --count );
            System.out.println(" " + count);
        }
}
```

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 **F(A,B,C,D) = Σ (0, 2, 3, 6, 8, 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 **F(P,Q,R,S) = π (5, 7, 8, 10, 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]**

Question 5

(a) Draw the logic diagram and truth table to encode the decimal numbers (**2, 3, 5, 7, 8**) and briefly explain its working. **[5]**

(b) Simplify the following Boolean expression and draw the gate for the reduced expression.

F= A'B + AB'C + A **[2]**

(c) Define *Universal gates*. Give one example and show how it works as an OR gate. **[3]**

Question 6

(a) Draw a truth table with a 3 input combination which outputs 1 if there are odd number of 0's. Also derive an **SOP** expression for the output. Reduce the expression using Karnaugh Map. **[5]**

(b) Define *Proposition*. How does tautology differ from contradiction? **[2]**

(c) Briefly explain the working of a 4:1 multiplexer. Also draw the logic diagram of 4:1 multiplexer. **[3]**

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 **[10]**

A class **Composite** contains a two dimensional array of order [m x n]. The maximum values possible for both 'm' and 'n' is 20. Design a class **Composite** to fill the array with the first (m x n) composite numbers in column wise. The details of the members of the class are given below:

Class name **: Composite**

Data members /instance variables :

arr[ ][ ]	: stores the composite numbers column wise
m	: integer to store the number of rows
n	: integer to store the number of columns

Member functions :

Composite(int mm, int nn)	: to initialize the size of the matrix m=mm and n=nn
int isComposite(int p)	: returns 1 if number is composite otherwise returns 0.
void fill ( )	: to fill the elements of the array with the first (m × n) composite numbers in column wise
void display( )	: displays the array in a matrix form.

Specify the class **Composite** giving details of the **constructor(int,int)**, **int isComposite(int), void fill( )** and **void display( ).** Define a **main( )** function to create an object and call the functions accordingly to enable the task.

Question 8 **[10]**

Design a class **Sort** which enables a word to be arranged in alphabetical order. The details of the members of the class are given below :

Class name : Sort

Data members /instance variables:

Str	: stores a word
len	: to store the length of the word

Member functions :

Sort( )	: default constructor
void readword( )	: to accept the word
void arrange ( )	: to arrange the word in alphabetical order using any standard sorting technique.
void display( )	: displays the original word along with the sorted word

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

Question 9 **[10]**

A Special number is a number in which the sum of the factorial of its digits is equal to the number. Example: 145 (1! + 4! + 5! = 145). Thus, 145 is a special number.
Design a class **Special** to check if the given number is a Special number or not. Some of the members of the class are given below:

Class name : Special

Data members /instance variables :

n	: integer to store the number

Member functions :

Special( )	: default constructor
void read( )	: to accept the number
int factorial(int x)	: return the factorial of a number using **recursion technique**.
boolean isSpecial( )	: checks for the special number by invoking the function factorial( ) and returns true if Special, otherwise returns false
void display( )	: to show the result with an appropriate message.

Specify the class **Special**, giving details of the **Constructor**, **void read( ), int factorial(int), boolean isSpecial( ) and void display( )**. 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 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 comments in the program and mnemonic names or pseudo codes for algorithms. The program must be written in Java and the algorithms must be written in general / standard form, wherever required / specified.
(Flowcharts are **not** required.)

Question 10 **[5]**

A super class **Account** contains employee details and a sub class **Simple** calculates the employee's simple interest. The details of the two classes are given below:

Class name	**:**	**Account**
Data Members:		
Name	:	stores the employee name
Pan	:	stores the employee PAN number
Principal	:	stores the Principal amount (in decimals)
acc_no	:	stores the employee bank account number
Member functions:		
Account(….)	:	parameterized constructor to assign value to data members
void display( )	:	to display the employee details

Class name	**:**	**Simple**
Data Members:		
time	:	stores the time duration
rate	:	stores the rate of interest
interest	:	stores the simple interest
Member functions:		
Simple(….)	:	parameterized constructor to assign value to data members of both the classes.
void calculate( )	:	calculates the simple interest as (Principal × time × rate) / 100
void display( )	:	displays the employee details along with the rate, time and interest.

*Assume that the super class **Account** has been defined.* Using the **concept of inheritance**, specify the class **Simple** giving details of **constructor**, **void calculate( )** and **void display( )**. **The super class and the main function need not be written**.

Question 11 **[5]**

A dequeue enables the user to add and remove integers from both the ends i.e. front and rear. Define a class **DeQueue** with the following details:

Class name : DeQueue

Data Members:

ele[ ]	:	array to hold the integer elements.
cap	:	stores the maximum capacity of the array.
front	:	to point the index of the front.
rear	:	to point the index of the rear.

Member functions:

DeQueue(int max)	:	constructor to initialize the data member cap = max, front = rear = 0 and create the integer array
void pushfront(int v)	:	to add integers from the front index if possible else display the message("full from front").
int popfront( )	:	to remove the return elements from front. If array is empty then return-999.
void pushrear(int v)	:	to add integers from the front index if possible else display the message("full from rear").
int poprear( )	:	to remove and return elements from rear. If the array is empty then return-999.

Specify the class **DeQueue** giving the details of **ONLY** the **constructor(int)**, **void pushfront(int)** and **int poprear ( )**. Assume that the other functions have been defined.

The main( ) function need not be written.

Question 12

(a) A linked list is formed from the objects of the class: **[2]**

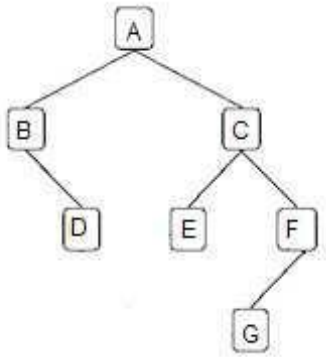
```
class Node
{
        int num;
        Node next;
}
```

Write an *Algorithm* **OR** a *Method* to insert a node at the beginning of an existing linked list.

The method declaration is as follows:

void InsertNode(Nodes starPtr, int n)

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



(i) Name the Root and the leaves of the tree. **[1]**

(ii) Write the post order traversal of the tree. **[1]**

(iii) Separate the Internal nodes and the External nodes of the tree. **[1]**

PLEASE NOTE: The total weightage of questions in the Question Paper will be as indicated in the Specimen Paper. However, breakup of subparts in questions may vary from one year to another.

COMPUTER SCIENCE

Paper – 2

(PRACTICAL)

(Three hours)

Maximum Marks: 30

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

The total time to be spent on the Planning session and Examination session is Three hours.

Planning session: 90 minutes

Examination session: 90 minutes

Note: Candidates are to be permitted to proceed to the Examination Session only after the 90 minutes of the Planning Session are over.

---

*This paper consists of **three** problems from which candidates are required to attempt **any one** problem.*

Candidates are expected to do the following:

1. Write an algorithm for the selected problem.
   (Algorithm should be expressed clearly using any standard scheme such as pseudo code or in steps which are simple enough to be obviously computable) **[3]**

2. Write a program in **JAVA** language. The program should follow the algorithm and should be logically and syntactically correct. **[5]**

3. Document the program using mnemonic names / comments, identifying and clearly describing the choice of data types and meaning of variables. **[2]**

4. Code / Type the program on the computer and get a print out (Hard Copy). Typically, this should be a program that compiles and runs correctly. **[2]**

5. Test run the program on the computer using the given sample data and get a print out of the output in the format specified in the problem. **[5]**

6. Viva-Voce on the **Selected Problem**. **[3]**

In addition to the above, the practical file of the candidate containing the practical work related to programming assignments done during the year is to be evaluated as follows:

- Programming assignments done throughout the year (by the Teacher) **[5]**
- Programming assignments done throughout the year (by the Visiting Examiner) **[5]**

*Solve any **one** of the following Problems.*

Question 1

An Evil number is a positive whole number which has even number of 1's in its binary equivalent.

Example: Binary equivalent of 9 is 1001, which contains even number of 1's.

A few evil numbers are 3, 5, 6, 9....

Design a program to accept a positive whole number and find the binary equivalent of the number and count the number of 1's in it and display whether it is a Evil number or not with an appropriate message.

Output the result in format given below:

Example 1

INPUT : 15

BINARY EQUIVALENT : 1111

NO. OF 1's : 4

OUTPUT : EVIL NUMBER

Example 2

INPUT : 26

BINARY EQUIVALENT : 11010

NO. OF 1's : 3

OUTPUT : NOT AN EVIL NUMBER

Question 2

The encryption of alphabets are to be done as follows:

A = 1
B = 2
C = 3
.
.
.
Z = 26

The potential of a word is found by adding the encrypted value of the alphabets.

Example: KITE
Potential = 11 + 9 + 20 + 5 = 45

Accept a sentence which is terminated by either " . " , " ? " or " ! ". Each word of sentence is separated by single space. Decode the words according to their potential and arrange them in ascending order.

Output the result in format given below:

Example 1

INPUT : THE SKY IS THE LIMIT.

POTENTIAL :
THE = 33
SKY = 55
IS = 28
THE = 33
LIMIT = 63

OUTPUT :IS THE THE SKY LIMIT

Example 2

INPUT : LOOK BEFORE YOU LEAP.

POTENTIAL :
LOOK = 53
BEFORE = 51
YOU = 61
LEAP = 34

OUTPUT :LEAP BEFORE LOOK YOU

Question 3

Given a square matrix M [ ] [ ] of order 'n'. The maximum value possible for 'n' is 10. Accept three different characters from the keyboard and fill the array according to the instruction given below.

Fill the upper and lower elements formed by the intersection of the diagonals by character

1. Fill the left and right elements formed by the intersection of the diagonals by character 2.

Fill both the diagonals by character 3.

Output the result in format given below:

Example 1

ENTER SIZE : 4

INPUT : FIRST CHARACTER '*'
SECOND CHARACTER '?'
THIRD CHARACTER '#'

OUTPUT :

```
# *   * #
? #   # ?
? #   # ?
# *   * #
```

Example 2

ENTER SIZE : 5

INPUT : FIRST CHARACTER '$'
SECOND CHARACTER '!'
THIRD CHARACTER '@'

OUTPUT :

```
@  $  $  $  @
!  @  $  @  !
!  !  @  !  !
!  @  $  @  !
@  $  $  $  @
```

Example 3

ENTER SIZE : 65

OUTPUT :SIZE OUT OF RANGE
