ANNUAL EXAMINATION 2018-19

Class XI (ISC)

COMPUTER SCIENCE Paper - I (Theory)

Time: Three hours **Maximum marks: 70**

Instructions:

* *Answers to this paper must be written on the answer script provided separately.*
* *You will **not** be allowed to write during the first 15 minutes. The time is to be spent in reading the question paper.*
* *All subsections of each question must be answered in the correct order.*
* *All working including rough work should be done on the same sheet as the rest of the answer.*
* *Please do not write anything on the question paper except your name and roll number.*
* *The intended marks for questions or parts of questions are given in brackets [ ].*
* *Answer **all** the questions from **Part I** and **six** questions from **Part II** choosing **two** questions from **Section A**, **two** questions from **Section B** and **two** questions from **Section C**.*

PART I [20 marks]

*Attempt **all** questions from this Part.*

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

Question 1

(a) Draw a truth table to represent a two input XNOR gate. [1]

(b) Verify if, A′ + A.B = A′ + B [1]

(c) What is the difference between simple and compound proposition? [1]

(d) Minimize: F = A.B.C′ + A.B.C + A.B′ [1]

(e) If A denotes "I can sing" and B denotes "I can dance", express the following statement in symbolic form: [1]

(i) If I can sing then I can dance.

(ii) If and only if I can dance then I can sing.

Question 2

(a) What is the difference between recursion and iteration? [2]

(b) What is the significance of keyword this? Give an example. [2]

(c) How are exceptions handled in Java? [2]

(d) What is the difference between equals( ) and compareTo( ) function? [2]

(e) If String nm= "INDIA" then what will be printed by the statement:

```
System.out.println(nm.indexOf("Z") + nm.lastIndexOf("I"));
```

[2]

Question 3

(a) What will be the output after the given code executes? Show proper dry run/working:

```
int i;
for(i=1; i<=4; i++)
{
    System.out.println((i*2) + "    " + (3*(6–i)));
}
```

[2]

(b) What will be the output after the given code executes if the value of n = 3? Show proper dry run/working:

```
void find(int n)
{
    if(n = = 1)
    {System.out.println(n);
    }
    else
    {
        System.out.println(n);
        find(n–1);
        System.out.println(n);
    }
}
```

[3]

PART II [50 marks]

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

SECTION A [20 marks]

*Attempt **any two** questions from this Section.*

Question 4

(a) Differentiate between compile-time and run-time error. [2]

(b) Draw OR gate operation using NOR gates only (with proving). [2]

(c) Find out the equivalent boolean expression for $(p \leftrightarrow q) + (p \rightarrow q)$, without having $\rightarrow$ or $\leftrightarrow$ operator. [2]

(d) If $X \rightarrow Y'$ then find its: [2]

(i) Converse

(ii) Inverse.

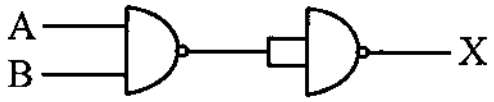
(e) Name the following: [2]

(i) an overloaded method of String class

(ii) a method that converts a String into a double data.

Question 5

(a) Given the following circuit:



Showing the working write the output X when:

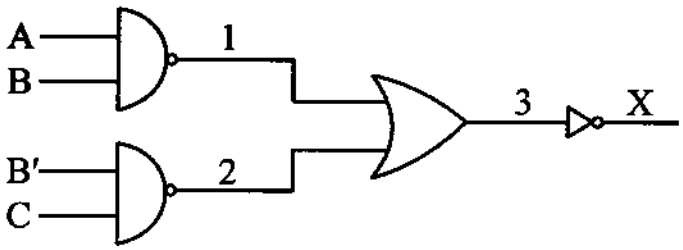
(i) A = 0, B = 0

(ii) A = 1, B = 0. [2]

(b) From p∧q and ~p infer q. State the laws used at each step. [2]

(c) Give two advantages of Exception handling. [2]

(d) From the given logic gate diagram,



find the expression for outputs '1', '2', '3' and 'X'. Simplify the expression X using Boolean laws. [4]

Question 6

(a) What is the difference between Half Adder and Full Adder? Make truth table and logic circuit of Full Adder. [5]

(b) Simplify the following using laws of Boolean algebra: [3]

a.(a′ + b) + b.(a′ + b′) + b′

(c) Which gates are called Universal gates and why? Draw a NOT gate using any of the universal gates. [2]

SECTION B [20 marks]

*Attempt **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.)

Question 7

Design a class Straw with the following details:

Class name	:	Straw
Data member / instance variable:		
str	:	to store any sentence.
Member functions / methods:		
Straw(String x)	:	initialize str by *x*
boolean hasVowel(String w)	:	returns true if the word in w contains one or more number of vowels otherwise returns false
void display( )	:	prints those words, of the sentence str, which contain one or more vowels, also counts and prints the number of words of str that do not contain any vowel.

Specify the class Straw giving details of the constructor and other methods. Write the main function to create the object of the class and call the methods accordingly. [10]

Question 8

Design a class MAT with the following details:

Class name	:	MAT
Data members / instance variables:		
A[]	:	integer array
n	:	size of array.
Member functions / methods:		
MAT(int nn)	:	constructor to initialize the size n by nn and to create the array
void readdata( )	:	accepts and fills the elements of the array in ascending order
void common(MAT B)	:	prints the common elements in array of object B and in array of the current object.

Specify the class MAT giving details of the constructor, void readdata( ) and void common(MAT). Write the main( ) function to create the objects and call the methods accordingly. [10]

Question 9

A Duck number is the one which contains at least one zero as one of its digits. For example 5048, 2020 etc. A class Ducknum is defined as follows:

Class name : Ducknum

Data member / instance variable:

num : to store an integer.

Member functions / methods:

Ducknum(int nx)	:	parameterized constructor to initialize num
int countzero(int n)	:	to count and return the number of zeroes in the argument using **recursive** technique
void checkDuck( )	:	checks whether the given number is a duck number or not using method countzero( ) and displays a suitable message.

Specify the class Ducknum giving details of the constructor and methods. Write the main method to create an object and call the methods accordingly. [10]

SECTION C [10 marks]

*Attempt **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.

(Flowcharts are not required.)

Question 10

To calculate the sum of the two diagonals of a square matrix a class DDA has been defined with the following details:

Class name : DDA

Data members / instance variables:

n[] []	: integer array
m	: size of the array (m × m).

Member functions / methods:

DDA(int size)	: constructor to assign m = size
void fillarray( )	: to accept integers in n[] []
int sumleft( )	: returns the sum of the elements in left diagonal of n[] []
int sumright( )	: returns the sum of the elements in right diagonal of n[] [].

Specify the class DDA giving details of the constructor and methods int sumleft( ) and int sumright( ) only. You need not write the main method. [5]

Question 11

A spy number is the number in which sum of its digits is equal to the product of its digits.

For example 123 ($1 + 2 + 3 = 1 \times 2 \times 3$).

A class Spynum has been defined to check whether the number entered is a spy number or not.

Class name : Spynum

Data member / instance variable:

num : to store a number.

Member functions / methods:

Spynum(int nn)	:	constructor to initialize member variable num
int sumdigits( )	:	returns the sum of digits of the number in num
int prodigits( )	:	returns the product of digits of the number in num
void check( )	:	checks and prints whether the number in num is spy or not.

Specify the class Spynum giving details of the constructor and methods. You need not write the main method. [5]

Question 12

Write an algorithm to store 10 numbers in an array and arrange them in decreasing order using insertion sort technique. [5]