

COMPUTER SCIENCE
PAPER 1
(THEORY)

21001072

Maximum Marks: 70

Time Allotted: Three Hours

Reading Time: Additional Fifteen Minutes

Instructions to Candidates

1. You are allowed an **additional fifteen minutes** for **only** reading the question paper.
2. You must **NOT** start writing during the reading time.
3. This question paper has **12 printed pages**.
4. It is divided into **two parts: Part I and Part II**.
5. It has **eleven questions** in all.
6. **Part I** is compulsory and has **two questions**.
7. While attempting **Multiple Choice Questions** in **Part I**, you are required to **write only ONE option** as the answer.
8. **Part II** is divided into **three sections: A, B and C**.
9. **Each section in Part II** has **three questions**. **Any two questions** have to be attempted from each section.
10. The intended marks for questions are given in brackets [].

Instruction to Supervising Examiner

1. Kindly read **aloud** the Instructions given above to all the candidates present in the examination hall.

PART I – 20 MARKS

Answer *all* questions.

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

Question 1

- (i) The Boolean equations $p + p' = 1$ and $p \cdot p' = 0$ correspond to: [1]

- (a) Involution Law
- (b) Absorption Law
- (c) Commutative Law
- (d) Complement Law

- (ii) According to the Principle of Duality, the Boolean equation $A + B' \cdot C + A' \cdot B \cdot C' = (A' + 0) \cdot (B + 1)$ will be equivalent to: [1]

- (a) $(A \cdot B + C) \cdot (A' + (B + C')) = (A' \cdot 1) + (B \cdot 0)$
- (b) $(A \cdot B') + C \cdot (A' + B + C') = (A + 1) \cdot (B' + 0)$
- (c) $A \cdot (B' + C) \cdot (A' + B + C') = (A' \cdot 1) + (B \cdot 0)$
- (d) $(A \cdot B' \cdot C' + A \cdot B + A' \cdot C) = (B' \cdot 1) \cdot (A \cdot 0)$

- (iii) Study the given propositions and the statements marked Assertion and Reason that follow it. Choose the correct option based on your analysis. [1]

P = You submit your assignment on time

Q = You get full marks for submission

If $P \Rightarrow Q$ then,

Assertion: Its inverse in statement form is:

If you do not submit your assignment on time, then you will not get full marks for submission.

Reason: A conditional statement $P \Rightarrow Q$ can be expressed as $\sim P \vee Q$

- (a) Both Assertion and Reason are true and Reason is the correct explanation for Assertion.
- (b) Both Assertion and Reason are true but Reason is not the correct explanation for Assertion.
- (c) Assertion is true and Reason is false.
- (d) Both Assertion and Reason are false.

(iv) The complement of the Boolean expression $p' \cdot q' + c' + p' \cdot q$ is: [1]

- (a) $p + q \cdot c \cdot p + q'$
- (b) $(p' + q) \cdot c \cdot (p + q')$
- (c) $(p + q') \cdot c' \cdot (p + q')$
- (d) $p' \cdot q + c + p \cdot q'$

(v) The *worst-case* complexity for following code segment is: [1]

```
for (int i = 0; i < p; i++)  
{  
    a += Math.sqrt(i);  
}  
for (int j = 0; j < q; j++)  
{  
    b += Math.log(j + 1);  
}
```

- (a) $O(1)$
- (b) $O(p \times q)$
- (c) $O(p^2 + q^2)$
- (d) $O(p + q)$

(vi) Given below are two statements marked, Assertion and Reason. Read the two statements carefully and choose the correct option. [1]

Assertion: In Java, a subclass can override a method of its superclass by defining a method with the same name and same signature.

Reason: If the subclass calls the overridden method without using the super keyword, the subclass's own version of the method will get called, leading to recursion.

- (a) Both Assertion and Reason are true and Reason is the correct explanation for Assertion.
- (b) Both Assertion and Reason are true but Reason is not the correct explanation for Assertion.
- (c) Assertion is true and Reason is false.
- (d) Both Assertion and Reason are false.

- (vii) Consider the following code segment:

[1]

```
class Student
{
    static int totalStu = 0;
    String name;
    public Student(String n)
    {
        name = n;
        totalStu++;
    }
}
```

Identify the valid statement(s)?

- I. The variable totalStu keeps count of all Student objects created
- II. Each object of the class gets its own copy of totalStu
- III. The variable totalStu can be accessed using Student.totalStu

- (a) Only I and II
- (b) Only I and III
- (c) Only II and III
- (d) Only III

- (viii) Given below are two statements marked, Assertion and Reason. Read the two statements carefully and choose the correct option. [1]

Assertion: In Java, an abstract class is created for the purpose of inheritance.

Reason: An abstract class can be instantiated.

- (a) Both Assertion and Reason are true and Reason is the correct explanation for Assertion.
- (b) Both Assertion and Reason are true but Reason is not the correct explanation for Assertion.
- (c) Assertion is true and Reason is false.
- (d) Both Assertion and Reason are false.

- (ix) Consider the two propositions given below: [1]

x = You stay calm

y = You will succeed

If x implies y, then write its *Converse* statement.

- (x) Write the reduced expression of $F(A, B, C) = \pi(0, 1, 2, 3, 4, 5, 6, 7)$ [1]

Question 2

- (i) Convert the following *infix* notation to *prefix* form. [2]

$$(P + Q / R) * (S + T / U) / V$$

- (ii) An array `Arr[3.....8, -5.....5]` is stored **Row Major Wise**. The address of `Arr[5][-2]` is 5120. If each element occupies 4 bytes in the memory, find the **base address** of the array. [2]

- (iii) The following function `justDoIt()` is a part of some class. Assume *num* is a positive integer.

```
public int justDoIt (int num)
{
    if (num == 0)
        return 0;
    int c = justDoIt (num / 2);
    int bit = num % 2;
    System.out.print(bit);
    return c + (bit == 1 ? 1 : 0);
}
```

Answer the questions given below along with dry run / working.

- (a) What will the function `justDoIt()` return when **num = 126**? [2]
(b) What is the function `justDoIt()` performing apart from recursion? [1]
- (iv) The following function `isNiven()` is a part of some class which is used to check if a given number is a Niven number or not. There are some places in the code marked by **?1?**, **?2?**, **?3?** which may be replaced by a statement / expression so that the function works properly.

A Niven number is a number which is exactly divisible by the sum of its digits.

Example: $120 = 1+2+0 = 3$ and 120 is exactly divisible by 3

```
boolean isNiven(int n)
{
    int sum = 0, temp = ?1?;
    while (temp > 0)
    {
        sum += ?2?;
        temp /= 10;
    }
    return (?3?);
}
```

What are the expressions or statements at **?1?**, **?2?** and **?3?** [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

Answer any two questions.

Question 3

- (i) (a) How many maximum output lines would be there in a 4-input *decoder*? [1]
(b) Draw the logic gate diagram for 8 : 1 *multiplexer*. [3]
(c) State *any one* application of an *encoder*. [1]
- (ii) The *Modus Ponens* states that $(p \wedge (p \Rightarrow q)) \Rightarrow q$. Prove this using Boolean laws. [3]
- (iii) Given $F(P, Q, R) = \pi(0, 1, 4, 7)$ write its canonical form. [2]

Question 4

- (i) An automated parking gate uses four binary sensors. The parking gate should open under any one of the following conditions: [5]
 - Vehicle detected and parking slot available
 - OR**
 - Parking slot available and either payment confirmed or VIP access granted
 - OR**
 - Payment not confirmed but VIP access granted

The inputs are:

INPUTS	
V	Vehicle detected
S	Parking slot available
P	Payment confirmed
I	VIP access granted

(In all the above cases, 1 indicates YES and 0 indicates NO)

Output: X - Denotes opening of the parking gate (1 = Open, 0 = Close)

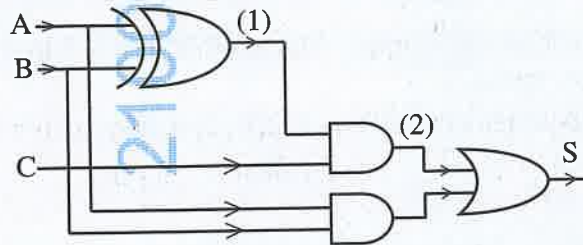
Draw the truth table for the inputs and outputs given above. Write the SOP expression for $X(V, S, P, I)$.

- (ii) Draw the logic gate diagram for a *Full Adder*. Write the Boolean expression for its SUM and CARRY. [3]
- (iii) Draw a truth table to verify if the following proposition is a Tautology, a Contradiction or a Contingency. [2]

$$(A \oplus B) = (A \odot B)'$$

Question 5

- (i) (a) Reduce the Boolean function $F(A, B, C, D) = \sum(0, 1, 2, 4, 8, 9, 10, 12)$ by using 4-variable Karnaugh map, showing the various groups (i.e., octal, quads and pairs). [4]
- (b) Draw the logic gate diagram for the reduced expression. Assume that the variables and their complements are available as inputs. [1]
- (ii) From the logic gate diagram given below, derive the Boolean expression for (1), (2) and S. Reduce the derived expression. [3]



- (iii) Draw the logic gate diagram for 2-input OR gate using NAND gates only. Show the expression at each step. [2]

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 6

- (i) Design a class **Powerful** to check if a given number is powerful or not. [8]
- A Powerful number is a number which is equal to the sum of its digits, raised to the power of the digit itself.
- Example: $3435 = 3^3 + 4^4 + 3^3 + 5^5 = 27 + 256 + 27 + 3125 = 3435$
- The details of the members of the class are given below:
- | | |
|---------------------------------------|--|
| Class name | : Powerful |
| Data member/instance variable: | |
| num | : to store a positive integer |
| Methods/Member functions: | |
| Powerful() | : constructor to initialise the data member with legal initial value |
| void input() | : to accept a positive integer |
| int sum(int num) | : to return the sum of the digit ^{digit} for all digits of num using recursive technique |
| void check() | : to check whether the given number is a powerful number by invoking the function sum() and display the result with an appropriate message |

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

- (ii) State the role of *base case* and *recursive case* in a recursive method. [2]

Question 7

[10]

A class **FibList** is designed to extract the terms of the Fibonacci series from a set of numbers.

A Fibonacci series starts with terms 0 and 1. All the other terms in this series are generated by adding the previous two terms.

[A Fibonacci series with 8 terms will be 0, 1, 1, 2, 3, 5, 8, 13]

Example: Input array = { 1, 4, 9, 3, 10, 13, 7 } then,

Output array = { 1, 3, 13 }

The details of the members of the class are given below:

Class name : **FibList**

Data members/instance variables:

list[] : array to hold positive integers

size : to store the size of array

Methods/Member functions:

FibList(int s) : constructor to assign *size = s*

void read() : to accept the elements of the array

boolean checkFib(int n) : to check and return true if *n* is a Fibonacci term otherwise return false.

FibList genFib(FibList fbl) : to return a new object containing only Fibonacci terms from the object *fbl* by invoking the method *checkFib()* as per the given instructions

void display() : to display the elements of the original array and the array holding the Fibonacci terms

Specify the class **FibList** giving the details of the **constructor()**, **void read()**, **boolean checkFib(int)**, **FibList genFib(FibList)** and **void display()**. Define the **main()** function to create objects and call the functions accordingly to enable the task.

Question 8

[10]

A class **Extract** is defined to get the text between the first and the last occurrences of a particular word from a sentence. In case the word occurs only once or does not occur at all, then the message gets displayed as "No such substring found".

Example: **Input:**

Sentence - It is not true that I do not like coffee

Word - not

Output: true that I do

The details of the members of the class are given below:

Class name : **Extract**

Data member/instance variable:

text : to store the sentence

Methods/Member functions:

Extract() : constructor to initialise the data member with legal initial value

void readSent() : to accept a sentence

String extractText(String wrd) : to extract and return the substring between the **first** and the **last** occurrence of *wrd* from *text*

void display() : to display the original sentence and the substring by invoking *extractText()*

Specify the class **Extract** giving the details of the **constructor()**, **void readSent()**, **String extractText(String)** and **void display()**. Define the **main()** function to create an object and call the functions 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 9

A class **CustomerService** is defined to resolve customer service requests in the order in which they are received.

The details of the members of the class are given below:

Class name : **CustomerService**

Data members/instance variables:

services[]	: array to store customer service request
size	: to store the maximum capacity of the array
first	: to store the index of the first customer service request
last	: to store the index of the last customer service request

Methods/Member functions:

CustomerService(int s)	: constructor to assign <i>size</i> = <i>s</i> , <i>first</i> = 0 and <i>last</i> = 0
void add(int reqst)	: to insert a request at index <i>last</i> , if space is available, otherwise display “Request cannot be accepted at the moment”
int del()	: to remove and return the request at index <i>first</i> , if any, else return -9999

(i) Specify the class **CustomerService** giving details of the functions **void add(int)** and **int del()**. Assume that the other functions have been defined. [4]

(ii) Name the entity described above and state its principle. [1]

Question 10

[5]

A superclass **Course** has been defined to store the basic details of a course. Define a subclass **Internship** to store internship-related information and calculate total earnings.

The details of both the members of the class are given below:

Class name : **Course**

Data members/instance variables:

title : to store the course title

duration : to store the course duration in months

Methods/Member functions:

Course(...) : parameterised constructor to assign values to its data members

void show() : to display the course details

Class name : **Internship**

Data members/instance variables:

company : to store company name

stipend : to store the monthly allowance

totalEarnings : to store the total earnings

Methods/Member functions:

Internship(...) : parameterised constructor to assign values to data members of both the classes

void calculate() : to calculate the total earnings as:
(stipend × duration)

void show() : to display course and internship details

Assume that the superclass **Course** has been defined. Using the **concept of Inheritance**, specify the class **Internship**, giving details of **constructor(...)**, **void calculate()** and **void show()**.

The super class, main function and algorithm need NOT be written.

Question 11

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

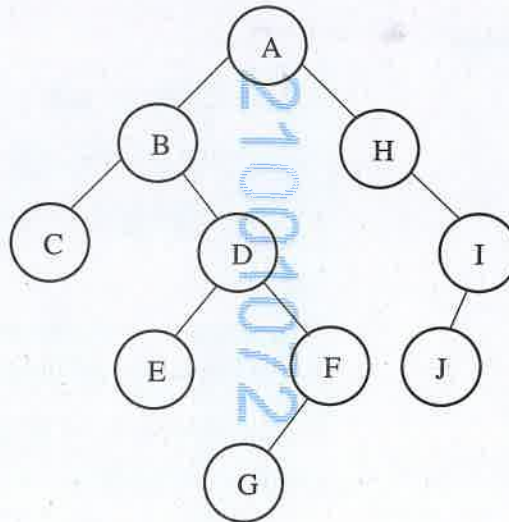
```
class Node
{
    char ch;
    Node next;
}
```

Write an *Algorithm* **OR** a *Method* to count and display the number of upper-case and lower-case characters.

The method declaration is as follows:

void countCase(Node startptr)

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



- (a) Write the *post-order* traversal of the above tree. [1]
(b) State the leaf nodes of the above tree. [1]
(c) State the parent of Node D and the sibling of Node E. [1]