VIT®

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

Final Assessment Test(FAT) - Apr/May 2025

Programme	**B.Tech.**	Semester	**Winter Semester 2024-25**
Course Code	**BCSE102L**	Faculty Name	**Prof. Sudha C**
Course Title	**Structured and Object-Oriented Programming**	Slot	**B1**
		Class Nbr	**CH2024250501633**
Time	**3 hours**	Max. Marks	**100**

Instructions To Candidates

- Write only your registration number in the designated box on the question paper. Writing anything elsewhere on the question paper will be considered a violation.

Course Outcomes

CO:Understand different programming language constructs and decision-making statements; manipulate data as a group.

CO2:Recognize the application of modular programming approach; create user-defined data types and idealize the role of pointers

CO3:Comprehend various elements of object-oriented programing paradigm; propose solutions through inheritance and polymorphism; identify the appropriate data structure for the given problem and devise solution using generic programming techniques.

Section - I

Answer all Questions (7 × 10 Marks)

01. Write a C program to analyze exam performance for 50 students, ensuring that scores are within the range of 0 to 100. The program should be structured using modular functions with proper parameter passing (no global variables).

The **readScores(int scores[])** function collects 50 valid scores from the user while ensuring the values are between 0 and 100. The **calculateAverage(const int scores[])** function computes and returns the average score while preventing division by zero. The **findMax(const int scores[])** function identifies and returns the highest score in the array, while the **findMin(const int scores[])** function determines and returns the lowest score. The **sortScores(int scores[])** function sorts the array in ascending. Finally, the **displayScores(const int scores[], float average, int max, int min)** function prints the sorted scores in a structured format, displaying 10 scores per line, along with the computed average, maximum, and minimum values. The program should ensure modularity, proper input validation, and efficient memory usage.

[10] (CO1/K3)

02. The LIC Insurance Premium System is designed to calculate and manage insurance premiums for multiple policyholders while optimizing performance using various storage classes and pointer arithmetic. A policyholder's monthly premium is stored using an auto variable, and a register variable is used to efficiently compute the total annual premium. To track the number of times premium calculations are performed, a static variable is utilized. Additionally, a global discount rate is defined as an extern variable, ensuring it can be accessed across multiple functions. The system also supports multiple policyholders by processing their premium data using pointer arithmetic. During testing, the system was given input for five policyholders, each with different monthly premiums. The program successfully computed their annual premiums, applied the discount rate, and displayed the total amount due for each policyholder.

(i) Write a function to calculate the total annual premium for a policyholder using a register variable. **[02 Marks]**

(ii) Implement a static counter to track and display the number of times the premium calculation function is called. **[02 Marks]**

(ii) Use an extern variable to apply a global discount rate to the total premium amount. **[02 Marks]**

(iv) Modify the program to allow processing of multiple policyholders' premiums using pointer arithmetic. **[02 Marks]**

(v) Analyze the effect of increasing the number of policyholders on the efficiency of the program, considering the impact of different storage classes. **[02 Marks]**

[10] (CO2/K3)

03. (i) Write a function **void rotate_array (int arr[], int size, int k, int direction);** using loops that rotates an array left or right by k positions using a loop. (direction = 1 → Right Rotation and direction = -1 → Left Rotation). The program should allow the user to enter the number of positions (k) and the direction of rotation. **[05 Marks]**

Main code:

```
#include <stdio.h>
void rotate_array(int arr[], int size, int k, int direction);
int main() {
  int arr[] = {1, 2, 3, 4, 5};
  int size = sizeof(arr) / sizeof(arr[0]);
  int k, direction;
  // User input
  printf("Enter number of positions to rotate: ");
  scanf("%d", &k);
  printf("Enter direction (1 for Right, -1 for Left): ");
  scanf("%d", &direction);

  printf("\nOriginal Array: ");
  for(int i = 0; i < size; i++)
    printf("%d ", arr[i]);

  rotate_array(arr, size, k, direction);
  printf("\nRotated Array: ");
  for(int i = 0; i < size; i++)
    printf("%d ", arr[i]);
  return 0;
}
```

(ii) **The following C program is intended to perform a search on an array to find a given target element. However, it contains five errors. Identify and correct them to ensure the program runs correctly. [05 Marks]**

```
#include <stdio.h>
int Search(int arr[], int n, int target) {
  for (int i = 0; i <= n; i++) {
    if (arr[i] = target) {
      return i;
    }
  }
  return 0;
}
int main() {
  int arr[] = {10, 20, 30, 40, 50};
  int size = sizeof(arr) / sizeof(arr[0]);
  int target = 30;
  int index = search(arr, size, target);
  if (index)
    printf("Element found at index %d\n", index);
  else
    printf("Element not found\n");

  return 0;
}
```

04. Develop a C++ program to simulate a **secure banking system using friend classes and static data members.** The system should include a **BankAccount class** that manages individual accounts by storing the **account holder's name and balance as private data members.** It should also use a static data member to keep track of the total number of accounts created. Additionally, the class should provide **functions for depositing and withdrawing** money, ensuring that all transactions adhere to predefined validation checks. **[10] (CO1/K4)**

The **BankManager class** should function as a friend class of **BankAccount,** allowing it to access private members of the account class. This class should provide functions to display the total number of accounts created and retrieve account details without modifying the account balance directly.

The main function should create objects for handling multiple users of the **BankAccount class** and perform various deposit and withdrawal operations while ensuring that all transactions meet the necessary validation requirements. It should also utilize the BankManager class to display account details and keep track of the total number of accounts in the system.

The system enforces several constraints to ensure secure and valid transactions. Each account must maintain a minimum balance of Rs. 1000, and any withdrawal that would bring the balance below this limit is not permitted. Additionally, the withdrawal amount cannot exceed the available balance, ensuring that no overdraft occurs.

To prevent excessive cash withdrawals, the maximum withdrawal limit per transaction is set at Rs. 5000. Furthermore, all deposits and withdrawals must be positive values, preventing invalid transactions. To encourage meaningful deposits, the system requires that each deposit be at least Rs. 500. These constraints collectively ensure financial stability and prevent unauthorized or erroneous transactions.

[10] (CO3/K3)

05. A group of 15 friends goes to a restaurant to celebrate the birthday of one among them. After enjoying their meals, the bill is generated. As per their tradition, the birthday person is exempted from paying, so the total bill is to be equally shared among the remaining 14 friends. Write a **C++ program using classes and objects** to model this scenario. The program should:
    - Prompt the **user to input the total bill amount.**
    - Use **two objects** of the class Fraction to represent the **numerator** (total bill) and the **denominator** (number of paying friends, i.e., 14).
    - Use a **parameterized constructor** to initialize the values.
    - **Overload the / operator** to perform division between the two Fraction objects.
    - Include methods to:

    a. Display the fraction in the format X/Y. (5 Marks)

    b. Display the result as a **floating-point value** using the overloaded / operator. (5 Marks)

    Ensure all data members are declared **private** and the program demonstrates **operator overloading** effectively.

    [10] (CO3/K5)

06. Develop a C++ program to extract unique elements from two datasets using **a function template**. The dataset consists of one integer array and one float array with a threshold condition (e.g., greater than a value). Implement a function template **'uniqueWithThreshold'** that takes a size n, a threshold value, and elements of two datasets, then outputs unique elements exceeding the threshold in order on separate lines.

    [10] (CO3/K3)

07. Imagine that you're working in a drone manufacturing company as a software developer. You're assigned the role of designing a real-time altitude tracking system for a drone. In a real drone, altitude data would come from onboard sensors (e.g., barometer, GPS, or ultrasonic sensors), but since this is a simulation, you will emulate that behavior in software.

    (i) Define a DroneAltitudeTracker class with private data (using std::list to store altitudes and a constant for the maximum size of 5) and public methods for adding (addAltitude()) and displaying altitudes (displayAltitudes()). **[03 Marks]**

    (ii) addAltitude() - Records a new altitude reading and ensures only the last 5 are kept. **[05 Marks]**

    Sample: [10.5, 15.2, 20.8, 25.0, 30.7] (size = 5).

    Add 28.9: Check size ($5 \geq 5$), remove 10.5 → [15.2, 20.8, 25.0, 30.7], add 28.9 → [15.2, 20.8, 25.0, 30.7, 28.9].

    (iii) displayAltitudes() - display the recent altitude history when requested. **[02 Marks]**

    Recent Altitude History (meters, oldest to newest): -

    15.2 m
    20.8 m
    25.0 m
    30.7 m
    28.9 m

    Create a C++ program using STL (built-in container list) to simulate this DroneAltitudeTracker.

    [10] (CO3/K3)

Section - II
Answer all Questions (2 × 15 Marks)

08. The **Storm Watch System** is designed to classify hurricanes and analyze wind speed variations using a C program with **pointers to structures.** The program should define a structure Hurricane containing attributes for **wind speed and category.** A function should determine the hurricane category based on predefined wind speed

ranges using a pointer to the structure. The program should allow users to input wind speeds for multiple hurricanes, classify each one accordingly, and display the results in a structured format. Additionally, **a separate function should compute the average wind speed of all recorded hurricanes**. Proper validation must be implemented to reject negative wind speeds. The program should utilize an array of structure pointers to store multiple hurricane records and perform necessary computations.

Hurricane ID	Wind Range (mph)	Category
1	74-95	I
2	96-110	II
3	111-130	III
4	131-155	IV
5	156 and above	V

Input :
Enter the number of hurricanes (max 100): 3
Enter wind speed (mph) for Hurricane 1: 110
Enter wind speed (mph) for Hurricane 2: 95
Enter wind speed (mph) for Hurricane 3: 135
Expected Output:
Hurricane Data:

ID	Wind Speed (mph)	Category
1	110	2
2	95	1
3	135	4

Average Wind Speed: 113.33 mph

[15] (CO2/K3)

09. Develop a C++ program for a university system to manage faculty members who lead research projects. **Use hybrid inheritance to model the relationships.** A base class Person has attributes name (string) and id (int). Derive a class Faculty from Person with an attribute department (string). Create another base class Researcher with an attribute researchArea (string). Derive a class ProjectLead that inherits from both Faculty and Researcher, adding attributes projectName (string), teamSize (int), and a dynamically allocated array teamMembers (string*) to store team member names. Include a method to add team members. Ensure proper memory management. The ProjectLead should calculate the project funding based on the team size: funding = teamSize * 5000 + 10000 (in dollars). Display all details, including the calculated funding, ensuring no ambiguity in the inheritance hierarchy. Write a main() function to create a ProjectLead object, take user input for its attributes, add team members based on the specified team size, and display the details with the funding. Give a neat sketch of the above said inheritance hierarchy.

Input :
Enter name: Dr. John Smith
Enter ID: 101
Enter department: Computer Science
Enter research area: Artificial Intelligence
Enter project name: AI for Healthcare
Enter team size: 2
Enter team member 1: Alice
Enter team member 2: Bob
Expected Output:
Name: Dr. John Smith, ID: 101
Department: Computer Science
Research Area: Artificial Intelligence
Project Name: AI for Healthcare
Team Members: Alice, Bob
Team Size: 2
Project Funding: $20000

[15] (CO3/K6)

BL-Bloom's Taxonomy Levels - (K1-Remembering,K2-Understanding,K3-Applying,K4-Analysing,K5-Evaluating,K6-Creating)