Lab: 02 (Comparative Practice of Union, Structure, and Class in C++)

Problem 1: Exploring Unions

Task:

1. Create a union called SensorData that can store either:
    - An integer temperature (in Celsius),
    - A float humidity (as a percentage),
    - A char array status (with a size of 10).
2. Write a C++ program that:
    - Initializes a SensorData variable.
    - Assigns a value to the temperature member and prints it.
    - Assigns a value to the humidity member and prints it.
    - Assigns a string to the status member and prints it.

Expected Output: You should observe that only the last assigned member holds a valid value, and previous values are overwritten.

Problem 2: Working with Structures

Task:

1. Define a structure called Student that contains:
    - An integer id to store the student ID.
    - A character array name (with a size of 50) to store the student's name.
    - A float gpa to store the student's GPA.
2. Write a C++ program that:
    - Creates an array of 3 Student objects.
    - Prompts the user to input the details for each student (ID, name, and GPA).
    - Prints the details of each student using a loop.

Expected Output: The program should display the details of all students, with each student's data stored separately and not overwritten.

Problem 3: Creating a Simple Class

Task:

1. Create a class called Book with the following private members:
    - An integer id.
    - A character array title (with a size of 100).
    - A float price.
2. The class should have public methods:
    - setDetails(int, const char*, float) to set the book's ID, title, and price.
    - display() to print the book's details.
3. Write a C++ program that:
    - Creates an object of the Book class.
    - Sets the details of the book using the setDetails method.
    - Displays the book's details using the display method.

Expected Output: The program should correctly display the book's details, demonstrating encapsulation.

Problem 4: Comparing Memory Usage

Task:

1. Create a structure called EmployeeStruct and a class called EmployeeClass that both contain the following members:
    - An integer id.
    - A character array name (with a size of 50).
    - A float salary.
2. Write a C++ program that:
    - Instantiates an object of both EmployeeStruct and EmployeeClass.
    - Uses the sizeof() function to print the size of each object.
3. Modify the program to include additional members (e.g., department, position) and compare the size again.

Expected Output: The program should demonstrate that structures and classes generally have similar memory usage, but classes may have additional overhead due to features like access control and member functions.

Problem 5: Real-World Scenario Simulation

Task:

1. Simulate a simple banking system using classes. Create a class called BankAccount with the following private members:
    - A string accountHolderName.
    - An integer accountNumber.
    - A double balance.
2. The class should have public methods:
    - deposit(double amount) to add money to the account.
    - withdraw(double amount) to withdraw money (with a check to ensure the balance doesn't go negative).
    - display() to show the account details.
3. Write a C++ program that:
    - Creates multiple BankAccount objects.
    - Performs deposit and withdrawal operations on each account.
    - Displays the final state of each account.

Expected Output: The program should correctly handle deposits, withdrawals, and display the final balance, showcasing the benefits of encapsulation and class-based design.

Submission Requirements:

- For each problem, submit the C++ source code **(Handwritten also)** file with clear comments explaining your logic.
- Ensure that your programs are properly tested and provide the expected outputs.
- Analyze and compare the differences in behavior between unions, structures, and classes in your lab report.