C++ Classes & Friend Functions Programming Homework

Date: July 29, 2025

Subject: Object-Oriented Programming - Classes, Objects & Friend Functions

Total Questions: 1

Instructions:

- Write a complete C++ program implementing the specified requirements
- Include proper header files and namespace declarations
- Add detailed comments to explain your logic
- Use proper class design principles
- Test your program with sample data for both subjects
- Submit well-formatted code with proper indentation

PROGRAMMING QUESTION

Question: Student Test Marks Management System using Classes and Friend Functions

Problem Statement: Write a C++ program to create two classes `Test1` and `Test2` which store marks of a student. Read values for class objects and calculate the average of two tests using a friend function. The program should handle marks for two subjects: **OOP (Object-Oriented Programming)** and **DBMS (Database Management System)**.

Technical Requirements:

Class Structure Implementation:

- **Encapsulation Paradigm:** Create two separate classes with private data members for marks storage
- **Friend Function Mechanism:** Implement inter-class data access using friend function declarations
- **Polymorphic Input Handling:** Design methods to accept marks for multiple subjects

In Simple Terms:

- **Two Separate Classes:** Make two classes that can store test marks privately
- **Friend Function:** Create a special function that can access private data from both classes to calculate averages
- **Multiple Subjects:** Handle marks for both OOP and DBMS subjects

---

Detailed Requirements:

Class Design:

1. **Test1 Class:**
    - Private members to store marks for OOP and DBMS
    - Public methods to input marks
    - Declare friend function for average calculation
2. **Test2 Class:**
    - Private members to store marks for OOP and DBMS
    - Public methods to input marks
    - Declare friend function for average calculation
3. **Friend Function:**
    - Calculate average marks for each subject across both tests
    - Display results in a formatted manner

Program Features:

- Input validation for marks (0-100 range)
- Clear user interface for data entry
- Formatted output showing individual test marks and averages
- Proper error handling for invalid inputs

Expected Program Structure:

```cpp
```

```
#include <iostream>
using namespace std;

class Test2; // Forward declaration

class Test1 {
    // Private data members for storing marks
    // Public methods for input/display
    // Friend function declaration
};

class Test2 {
    // Private data members for storing marks
    // Public methods for input/display
    // Friend function declaration
};

// Friend function definition
// Main function with object creation and testing
```

Sample Expected Output:

```
===== STUDENT TEST MARKS MANAGEMENT SYSTEM =====

Enter marks for Test 1:
OOP marks (0-100): 85
DBMS marks (0-100): 78

Enter marks for Test 2:
OOP marks (0-100): 92
DBMS marks (0-100): 88

===== RESULTS =====
Test 1 Marks:
OOP: 85, DBMS: 78

Test 2 Marks:
OOP: 92, DBMS: 88

===== AVERAGE CALCULATION =====
Average OOP marks: 88.5
Average DBMS marks: 83.0

Overall Average: 85.75
```

Implementation Guidelines:

Object-Oriented Design Principles:

1. **Data Encapsulation:** Keep marks as private members
2. **Information Hiding:** Access private data only through public methods and friend functions
3. **Class Cohesion:** Each class should handle its own test data
4. **Functional Coupling:** Friend function provides controlled access between classes

Programming Best Practices:

- Use meaningful variable names
- Include input validation
- Add proper comments explaining class relationships
- Handle edge cases (like invalid mark ranges)

- Format output for readability

Learning Objectives:

By completing this homework, you will master:

Advanced OOP Concepts:

- **Class Declaration and Definition:** Understanding blueprint creation for objects
- **Access Specifier Implementation:** Managing public, private data member visibility
- **Friend Function Paradigm:** Inter-class communication without inheritance
- **Object Instantiation:** Creating and manipulating class instances

In Simple Terms:

- **Making Classes:** How to create templates for storing student data
- **Keeping Data Safe:** How to hide important information and control who can see it
- **Special Friend Functions:** How to let specific functions access private information from multiple classes
- **Creating Objects:** How to make actual copies of your class templates to store real data

---

Submission Requirements:

1. **File Name:** `student_marks_system.cpp`
2. **Code Documentation:** Include header comments with your name, date, and program description
3. **Testing:** Test with at least 3 different sets of marks
4. **Error Handling:** Include validation for mark ranges (0-100)
5. **Formatting:** Use consistent indentation and spacing

Bonus Challenge (Optional):

Extend the program to:

- Handle more than 2 tests
- Add letter grade calculation based on average
- Store multiple students' data

- Calculate class average across all students

Good Luck with your Object-Oriented Programming practice!