CPP-A14: Access Modifiers, Friend Functions, and Friend Classes

C++ Implementation Problems

1. Implementing Access Modifiers in a Bank System

Problem:

Create a class `BankAccount` with:

- `private` members: `accountNumber`, `balance`
- `public` members:
  - Constructor to initialize account
  - `deposit(double amount)`: Adds money to balance
  - `withdraw(double amount)`: Withdraws money if sufficient balance
  - `displayBalance()`: Prints current balance
- Test this class in `main()`.

---

2. Friend Function for Distance Calculation

Problem:

- Define a class `Point` with `private` data members `x` and `y`.
- Create a friend function `calculateDistance(Point p1, Point p2)` that calculates the distance between two points using the formula:

  distance=(x2−x1)2+(y2−y1)2distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}
- Implement the `main()` function to demonstrate the function.

---

3. Friend Class for Employee Details

Problem:

- Define a class `Employee` with:
  - `private` members: `name`, `salary`
  - Constructor to initialize values
- Define a `friend class` `HR` with a function `showSalary(Employee e)` that prints employee details.
- Demonstrate the working in `main()`.

---

4. Friend Function for Comparing Two Objects

Problem:

- Create a class `Box` with `private` members `length`, `width`, and `height`.
- Write a friend function `isBigger(Box b1, Box b2)` that compares the volume of two `Box` objects and returns the larger one.
- Demonstrate the function in `main()`.

---

5. Implementing Protected Inheritance in a School System

Problem:

- Define a class `Person` with:
  - `protected` members: `name`, `age`
  - A constructor to initialize values
- Derive a `Student` class that inherits `Person`.
- `Student` should have additional `private` members: `rollNo`, `grade`.
- Create a `public` method `display()` to print student details.
- Demonstrate the use of inheritance and `protected` access modifier.

---

6. Friend Function with Operator Overloading

Problem:

- Define a class `Complex` with `private` members `real` and `imaginary`.
- Implement a friend function to overload the `+` operator to add two `Complex` objects.

- Demonstrate operator overloading in `main()`.

---

7. Friend Function to Swap Private Members of Two Classes

Problem:

- Define two classes `ClassA` and `ClassB`, each with a `private` integer member.
- Write a `friend function swapValues()` that swaps the values of private members of both classes.
- Demonstrate the function in `main()`.

---

8. Using Friend Class in a Car System

Problem:

- Create a class `Car` with:
  - `private` data members: `brand`, `price`
  - Constructor to initialize values
- Create a `friend class` `Showroom` with a function to display private details of `Car`.
- Implement `main()` to test the working.

---

9. Implementing a Secure ATM System Using Access Modifiers

Problem:

- Define a class `ATM` with:
  - `private` members: `pin`, `balance`
  - `public` methods:
    - `setPin(int p)`: Sets a PIN
    - `withdraw(int amount, int enteredPin)`: Withdraws money only if correct PIN is entered
    - `displayBalance(int enteredPin)`: Displays balance only if PIN is correct
- Demonstrate secure access control.

---

10. Friend Function for Arithmetic Operations on Two Numbers

Problem:

- Define a class `Number` with:
  - `private` members `num1`, `num2`
  - Constructor to initialize values
- Write a `friend function` that performs addition, subtraction, multiplication, and division on these numbers.
- Display the results in `main()`.

Happy Coding!