## **C++ Programming Assignments - Week 4**

**Instructions:** Complete the following programming exercises. For each question, write a C++ program that demonstrates the concepts learned.

### **Question 1: Static and Non-Static Class Members**

1. Create a C++ class named Student.
2. The Student class should have the following members:
   * A non-static data member: std::string studentName.
   * A non-static data member: int studentID.
   * A static data member: int totalStudents. This member should keep track of the total number of Student objects created.
3. Implement a constructor for the Student class that initializes studentName and studentID, and increments totalStudents each time a new Student object is created.
4. Add a non-static method void displayStudentInfo() that prints the studentName and studentID of a specific student object.
5. Add a static method void displayTotalStudents() that prints the value of totalStudents. Static member functions can access only static data members.
6. In your main function:
   * Create at least three Student objects.
   * Call displayStudentInfo() for each non-static object.
   * Call displayTotalStudents() using the class name (e.g., Student::displayTotalStudents()) to show the total count of students created.

### **Question 2: Static Member Functions Restrictions**

1. Create a C++ class named Counter with:
   * A private static integer member count.
   * A private non-static integer member id.
2. Implement a constructor that increments count and assigns it to id for each new object.
3. Add a public static member function void showCount().
4. Inside showCount(), attempt to print the id of an object. What happens when you try to compile? Explain in comments within your code why this specific action (accessing a non-static member from a static function) is not allowed.
5. Modify showCount() to correctly print only the static count member.
6. In main, create a few Counter objects and call Counter::showCount().

### **Question 3: Single Inheritance**

1. Define a base class named Vehicle.
2. The Vehicle class should have:
   * A protected attribute std::string brand. (Protected members are accessible within the same class and by derived classes ).
   * A public method void honk() that prints "Vehicle makes a sound!".
3. Define a derived class named Car that publicly inherits from Vehicle.
4. The Car class should have its own public attribute std::string model.
5. Add a public method void displayCarInfo() to the Car class that prints the brand (inherited from Vehicle) and model of the car.
6. In your main function:
   * Create an object of the Car class.
   * Set its brand (via Vehicle) and model.
   * Call honk() (inherited method) and displayCarInfo() (its own method) using the Car object.

### **Question 4: Multilevel Inheritance**

1. Define a base class named Animal with a public method void eat() that prints "Animal is eating.".
2. Define an intermediate derived class named Dog that publicly inherits from Animal.
3. The Dog class should have a public method void bark() that prints "Dog is barking.".
4. Define a final derived class named GoldenRetriever that publicly inherits from Dog.
5. The GoldenRetriever class should have a public method void fetch() that prints "Golden Retriever is fetching.".
6. In your main function:
   * Create an object of the GoldenRetriever class.
   * Call eat(), bark(), and Workspace() using the GoldenRetriever object. This demonstrates that GoldenRetriever inherits features from both Dog and Animal.

### **Question 5: Inheritance Activity - Base and Derived Classes**

1. Define a base class named Shape with a public method void getArea() that prints "Calculating area...".
2. Create a derived class named Rectangle that publicly inherits from Shape.
3. The Rectangle class should have private attributes double length and double width.
4. Add a public constructor to Rectangle that takes length and width as arguments and initializes them.
5. Override the getArea() method in the Rectangle class to calculate and print the actual area of the rectangle.
6. In your main function:
   * Create an object of the Rectangle class.
   * Call its getArea() method.
   * Call the inherited getArea() method (if you wish to explicitly call the base class version, use rectangleObject.Shape::getArea();).