Below are 7 separate assignments working with classes, objects, and the different concepts related to classes. They come with a general description of the class and then the specific details of the member functions and variables. Use the names and access modifiers (private or public) specified for the member functions and variables. Implement each function according to the description. Test all of the behaviors for each class in the main function through the use of objects.

The last two assignments involve inheritance and will take multiple classes to complete.

1. **Student Class:**
   - Description: Create a Student class that represents a student with member variables for name, roll number, and an array to store their marks in different subjects. Implement member functions to calculate the average marks, display the student's information, and set the marks for each subject. Create objects of the class for multiple students and demonstrate their usage.

   - Member Variables:
     - `name` (private): A string to store the name of the student.
     - `rollNumber` (private): An integer to store the roll number of the student.
     - `marks` (private): An array of integers to store the marks of the student in different subjects.
     - `numSubjects` (private): An integer to store the number of subjects.

   - Member Functions:
     - `Student()` (constructor, public): A constructor to initialize the Student object with default values.
     - `Student(name, rollNumber, numSubjects)` (overloaded constructor, public): An overloaded constructor to initialize the Student object with provided values.
     - `setMarks(marksArray)` (public): A function to set the marks for each subject.
     - `calculateAverageMarks()` (public): A function to calculate the average marks of the student.
     - `displayInformation()` (public): A function to display the student's information.

2. **Bank Account Class:**
   - Description: Design a BankAccount class with private member variables for account number and balance. Implement functions to deposit and withdraw money from the account, and display the account balance. Create objects for different accounts and perform transactions on them.

   - Member Variables:
     - `accountNumber` (private): An integer to store the account number.
     - `balance` (private): A double to store the account balance.

   - Member Functions:
     - `BankAccount()` (constructor, public): A constructor to initialize the BankAccount object with default values.
     - `BankAccount(accountNumber, initialBalance)` (overloaded constructor, public): An overloaded constructor to initialize the BankAccount object with provided values.
     - `deposit(amount)` (public): A function to deposit money into the account.
     - `withdraw(amount)` (public): A function to withdraw money from the account.
     - `displayBalance()` (public): A function to display the account balance.

3. **Social Media Analytics:**
   - Description: Create a SocialMediaPost class to store information about a social media post, including the username, content, and number of likes. Implement functions to display post details and calculate the average number of likes across multiple posts.

   - Member Variables:
     - `username` (private): A string to store the username of the social media post.
     - `content` (private): A string to store the content of the social media post.
     - `numLikes` (private): An integer to store the number of likes of the post.

   - Member Functions:
     - `SocialMediaPost()` (constructor, public): A constructor to initialize the SocialMediaPost object with default values.
     - `SocialMediaPost(username, content, numLikes)` (overloaded constructor, public): An overloaded constructor to initialize the SocialMediaPost object with provided values.
     - `displayPostDetails()` (public): A function to display the details of the social media post.

4. **Employee Management System:**
   - Description: Design an Employee class to store information about employees, including their name, employee ID, and salary. Implement functions to calculate the bonus for each employee based on their salary and display their information. Create objects for multiple employees and demonstrate their usage.

   - Employee Class:
     - Member Variables:
       - `name` (private): A string to store the name of the employee.
       - `employeeID` (private): An integer to store the ID of the employee.
       - `salary` (private): A double to store the salary of the employee.

     - Member Functions:
       - `Employee()` (constructor, public): A constructor to initialize the Employee object with default values.
       - `Employee(empName, empID, empSalary)` (overloaded constructor, public): An overloaded constructor to initialize the Employee object with provided values.
       - `calculateBonus()` (public): A function to calculate the bonus of the employee based on the salary.
       - `displayInfo()` (public): A function to display the employee's name, ID, salary, and bonus (if calculated).

   - EmployeeManagement Class:
     - Member Variables:
       - `employees` (private): An array of Employee objects to store the employee data.
       - `numEmployees` (private): An integer to keep track of the number of employees in the management system.

     - Member Functions:
       - `EmployeeManagement()` (constructor, public): A constructor to initialize the EmployeeManagement object with default values.
       - `EmployeeManagement(employeeArray, num)` (overloaded constructor, public): An overloaded constructor to initialize the EmployeeManagement object with provided employees and the number of employees.
       - `addEmployee(newEmployee)` (public): A function to add a new employee to the management system.
       - `removeEmployee(employeeID)` (public): A function to remove an employee from the management system based on their ID.
       - `displayAllEmployees()` (public): A function to display the details of all employees in the management system.

5. **Book and Library Classes:**
   - Description: Create classes for Book and Library. The Book class should have member variables for book title, author, and ISBN. Implement functions to display book details. The Library class should have an array to store books and functions to add, remove, and display books. Create objects for the Library class and demonstrate book management operations.

   - Book Class:
     - Member Variables:
       - `title` (private): A string to store the title of the book.
       - `author` (private): A string to store the author of the book.
       - `ISBN` (private): A string to store the ISBN (International Standard Book Number) of the book.

     - Member Functions:
       - `Book()` (constructor, public): A constructor to initialize the Book object with default values.
       - `Book(bookTitle, bookAuthor, bookISBN)` (overloaded constructor, public): An overloaded constructor to initialize the Book object with provided values.
       - `displayDetails()` (public): A function to display the details of the book, including title, author, and ISBN.

   - Library Class:
     - Member Variables:
       - `books` (private): An array of Book objects to store the collection of books.
       - `numBooks` (private): An integer to keep track of the number of books in the library.

     - Member Functions:
       - `Library()` (constructor, public): A constructor to initialize the Library object with default values.
       - `Library(bookArray, num)` (overloaded constructor, public): An overloaded constructor to initialize the Library object with provided books and the number of books.
       - `addBook(newBook)` (public): A function to add a new book to the library.
       - `removeBook(bookTitle)` (public): A function to remove a book from the library based on its title.
       - `displayAllBooks()` (public): A function to display the details of all books in the library.

6. **Shape Hierarchy:**
   - Description: Create a hierarchy of shapes using inheritance. Design a base class called "Shape," which contains member variables for the color and area. Implement member functions to calculate the area of the shape and display its details. Then, derive three classes (Circle, Rectangle, and Triangle) from the base "Shape" class. Each derived class should override the area calculation method according to its specific shape formula. Demonstrate the usage of these classes by creating objects for each shape, setting their properties, and displaying their information.

   - Base Shape Class:
     - Member Variables:
       - `color` (protected): A string to store the color of the shape.
       - `area` (protected): A double to store the area of the shape.

     - Member Functions:
       - `Shape()` (constructor, public): A constructor to initialize the shape's color with default values.
       - `Shape(shapeColor)` (overloaded constructor, public): An overloaded constructor to initialize the shape's color with the provided value.
       - `calculateArea()` (public, virtual): A virtual function to calculate the area of the shape.
       - `displayDetails()` (public): A function to display the shape's color and area.

   - Circle Class (Derived from Shape):
     - Member Variables:
       - `radius` (private): A double to store the radius of the circle.

     - Member Functions:
       - `Circle()` (constructor, public): A constructor to initialize the circle's radius with default values.
       - `Circle(shapeColor, circleRadius)` (overloaded constructor, public): An overloaded constructor to initialize the circle's color and radius with provided values.
       - `calculateArea()` (public, override): A function to calculate the area of the circle.
       - `displayDetails()` (public): A function to display the circle's color, radius, and area.

   - Rectangle Class (Derived from Shape):
     - Member Variables:
       - `length` (private): A double to store the length of the rectangle.
       - `width` (private): A double to store the width of the rectangle.

     - Member Functions:
       - `Rectangle()` (constructor, public): A constructor to initialize the rectangle's length and width with default values.
       - `Rectangle(shapeColor, rectLength, rectWidth)` (overloaded constructor, public): An overloaded constructor to initialize the rectangle's color, length, and width with provided values.

- `calculateArea()` (public, override): A function to calculate the area of the rectangle.
- `displayDetails()` (public): A function to display the rectangle's color, length, width, and area.

- Triangle Class (Derived from Shape):
  - Member Variables:
    - `base` (private): A double to store the base length of the triangle.
    - `height` (private): A double to store the height of the triangle.

  - Member Functions:
    - `Triangle()` (constructor, public): A constructor to initialize the triangle's base and height with default values.
    - `Triangle(shapeColor, triBase, triHeight)` (overloaded constructor, public): An overloaded constructor to initialize the triangle's color, base, and height with provided values.
    - `calculateArea()` (public, override): A function to calculate the area of the triangle.
    - `displayDetails()` (public): A function to display the triangle's color, base, height, and area.

7. **Vehicle Rental Management:**
   - Description: Create a base class called "Vehicle" to represent a vehicle with member variables for vehicle type, registration number, and rental rate. Implement member functions to display vehicle details and calculate the rental cost based on the number of days rented. Then, derive two classes (Car and Bike) from the base "Vehicle" class. Each derived class should override the rental cost calculation method according to its specific rental rates. Demonstrate the usage of these classes by creating objects for different vehicles, setting their properties, and displaying their rental cost.

   - Base Vehicle Class:
     - Member Variables:
       - `type` (protected): A string to store the type of the vehicle (e.g., "Car," "Bike").
       - `registrationNumber` (protected): A string to store the registration number of the vehicle.
       - `rentalRate` (protected): A double to store the rental rate per day.

     - Member Functions:
       - `Vehicle()` (constructor, public): A constructor to initialize the vehicle's type and registration number with default values.
       - `Vehicle(vehicleType, regNumber, ratePerDay)` (overloaded constructor, public): An overloaded constructor to initialize the vehicle's type, registration number, and rental rate with provided values.
       - `calculateRentalCost(numDays)` (public, virtual): A virtual function to calculate the rental cost based on the number of days rented.
       - `displayDetails()` (public): A function to display the vehicle's type, registration number, rental rate, and rental cost.

   - Car Class (Derived from Vehicle):
     - Member Variables:
       - `seats

` (private): An integer to store the number of seats in the car.

     - Member Functions:
       - `Car()` (constructor, public): A constructor to initialize the car's seats with default values.
       - `Car(vehicleType, regNumber, ratePerDay, numSeats)` (overloaded constructor, public): An overloaded constructor to initialize the car's type, registration number, rental rate, and seats with provided values.
       - `calculateRentalCost(numDays)` (public, override): A function to calculate the rental cost for the car based on the number of days rented.
       - `displayDetails()` (public): A function to display the car's type, registration number, rental rate, number of seats, and rental cost.

   - Bike Class (Derived from Vehicle):
     - Member Variables:

- `bikeType` (private): A string to store the type of the bike (e.g., "Standard," "Sports").

- Member Functions:
  - `Bike()` (constructor, public): A constructor to initialize the bike's type with default values.
  - `Bike(vehicleType, regNumber, ratePerDay, bikeType)` (overloaded constructor, public): An overloaded constructor to initialize the bike's type, registration number, rental rate, and bike type with provided values.
  - `calculateRentalCost(numDays)` (public, override): A function to calculate the rental cost for the bike based on the number of days rented.
  - `displayDetails()` (public): A function to display the bike's type, registration number, rental rate, bike type, and rental cost.

These assignments continue to build on the concepts of inheritance, classes, and member variables and functions while introducing new scenarios to explore and implement in C++.