Programming Assignment-2: Object-Oriented Programming in C++

Instructions:

- Implement each program in C++.
- Ensure proper use of classes, objects, and concepts mentioned in each question.
- Follow good coding practices, including comments and meaningful variable names.

---

1. Understanding Classes and Objects

Create a class `Student` with attributes `name`, `age`, and `grade`. Write a program to:

- Create an object of the `Student` class.
- Assign values to the attributes.
- Display the student details using a member function.

---

2. Constructors and Destructors

Create a class `Car` with attributes `brand`, `model`, and `year`.

- Implement a constructor to initialize these attributes.
- Implement a destructor that prints a message when an object is destroyed.
- Create objects inside and outside a block and observe when the destructor is called.

---

3. Dynamic Memory Allocation (`new` and `delete`)

Create a class `Book` with attributes `title` and `price`.

- Use dynamic memory allocation (`new` and `delete`) to create and delete an object of this class dynamically.
- Display book details before deallocating memory.

---

4. Function Overloading

Create a class `MathOperations` that has:

- A function `add(int, int)` to add two integers.
- A function `add(double, double)` to add two floating-point numbers.

- A function `add(string, string)` to concatenate two strings.
  Write a main function to call and test each overloaded method.

---

5. Friend Function

Create a class `Rectangle` with private attributes `length` and `width`.

- Implement a friend function `calculateArea()` that takes a `Rectangle` object as an argument and returns the area.
- Test the function by creating a `Rectangle` object and calling the friend function.

---

6. Pass by Value vs. Pass by Reference

Create a class `Number` with an integer attribute.

- Write a function `modifyValue()` that takes an object of `Number` by **value** and modifies its attribute.
- Write another function `modifyReference()` that takes an object of `Number` by **reference** and modifies its attribute.
- Show the difference in behavior when calling both functions.

---

7. Operator Overloading (`+` Operator)

Create a class `Complex` with attributes `real` and `imaginary`.

- Overload the `+` operator to add two `Complex` numbers.
- Test the overloaded operator in the `main()` function.

---

8. Operator Overloading (`==` Operator)

Create a class `Point` with `x` and `y` coordinates.

- Overload the `==` operator to compare two `Point` objects.
- Test the overloaded operator to check if two points are equal.

---

9. Overloading Unary `++` Operator

Create a class `Counter` with an integer attribute `value`.

- Overload the `++` operator (both pre-increment and post-increment).
- Demonstrate incrementing an object of `Counter`.

---

10. Constructor Overloading

Create a class `Person` with attributes `name` and `age`.

- Implement three constructors:
    1. Default constructor.
    2. Constructor with only `name`.
    3. Constructor with `name` and `age`.
- Demonstrate creating objects using all three constructors.

---

11. Friend Function with Two Classes

Create two classes `ClassA` and `ClassB`, each with a private integer attribute.

- Write a friend function `sumObjects()` that takes objects of both classes and returns their sum.
- Test the function in `main()`.

---

12. Operator Overloading (`<<` and `>>` for Input/Output Stream)

Create a class `Time` with attributes `hours` and `minutes`.

- Overload the `<<` operator to print the `Time` object.
- Overload the `>>` operator to take user input for `Time` object.
- Demonstrate these operators in `main()`.