Assignment 1:

A. Which problem is it ? What is/are the potential solution(s) to deal with this problem

- **The error occurs because title is a private member of the Book class.**

B. Explain why this one works (the program can be compiled)? Is it good to always use public specifier ?

- **works because publisher is declared as a public member of the Book class:**

C. What happens if we change from “public” to “private” for functions

- **Then, the main() function will not be able to call displayAuthorInfo() anymore because it's a private method.**

D. What are the access specifier/modifier ? Which are the benefits of private, public, and protected modifier E. What are classes in C++ and why do we need to use classes?

- **Private**: Prevents direct access to the class's internals, ensuring better control over how data is manipulated.
- **Public**: Allows access to class methods or members, facilitating interactions with the class in a controlled manner.
- **Protected**: Useful for inheritance, where derived classes need to access certain members of the base class, but you don’t want those members to be publicly available.

Assignment 2:

A. Explain why the program can be complied.

- All private attributes are accessed only via **public setter/getter methods**.
- All member functions and attributes are properly declared and defined.
- The `main()` function creates an object and uses the public interface correctly.

B. Which methods are good to set/modify/access private attributes.

- **Setters** (`setName`, `setAge`, `setGrade`) are used to **modify** private attributes from outside the class.
- **Getters** (`getName`, `getAge`, `getGrade`) are used to **access/read** the values of private attributes.

C. What does this line mean ? group (string) = “2025 group

- It means the **group** attribute is initialized with the default value "2025 group" at the time of object creation.

Assignment 3:

A. Explain what is constructor ? Why do we need to use constructor

A constructor is a special method in a class that is automatically called when an object is created. It is used to initialize the object's attributes.

Why use a constructor?

- To ensure that every object starts its life in a valid state.
- To set initial values for private attributes.

B. Explain what is destructor? Why do we need to use destructor

A destructor is a special method that is automatically called when an object goes out of scope or is explicitly deleted (if created with new). It cleans up resources like memory, file handles, etc.

Why use a destructor?

- To release resources.
- To avoid memory leaks.

C. Write the syntax for a constructor and a destructor for a class named Car. Can we return a value from constructor or destructor. Explain why ?

- No, constructors and destructors do not have return types, not even void.
- Their purpose is initialization and cleanup — not to return data.

D. What is the difference between creating an object using new vs creating it on the stack?

Key Differences:

- new allocates memory on the **heap** and must be manually deleted.
- Stack-allocated objects are **automatically** destroyed when they go out of scope.
- Use new when you need the object to **persist beyond the scope** it was created in.

E. What happens if you don't define a constructor in your class?

The compiler provides a **default constructor** if none is defined.

But if you define any constructor (like one with parameters), the default one is **not automatically created**.

If you try to create an object without providing arguments in such a case, you'll get a **compilation error**.

F. What happens if you don't define a destructor in your class ?

The compiler provides a **default destructor**.

It works fine unless you are managing **resources manually** (like dynamic memory, file handles, sockets, etc).

If your class uses dynamic memory and you don't define a destructor, you may cause **memory leaks**.

Assignment 4:

A. What is inheritance in C++? Explain how a derived class can access the members of its base class.

Inheritance in C++ is a mechanism where a new class (called a **derived class**) acquires the properties and behaviors (members) of an existing class (called a **base class**). It promotes **code reuse**, **extensibility**, and **polymorphism**.

A derived class can access base class members as follows:

- **Public and protected members** of the base class are **accessible** in the derived class.
- **Private members** are **not directly accessible** in the derived class, but they can be accessed via public or protected member functions of the base class.
- Access specifier (public, protected, or private) used during inheritance affects visibility.

B. In the ElectricCar class, why are there two constructors (a default constructor and a parameterized constructor)? What are the advantages of constructor overloading?

There are **two constructors** in ElectricCar:

- A **default constructor**: initializes the object with default values (e.g., batteryCapacity = 0).
- A **parameterized constructor**: allows initializing the object with specific values provided at runtime.

Advantages of constructor overloading:

- **Flexibility**: Users can create objects with or without passing parameters.
- **Convenience**: Reduces the need for manual member assignment after object creation.
- **Code clarity**: Multiple constructors serve different use cases clearly.

C. What would happen if the constructor of the base class (Vehicle or Car) is not explicitly called in a derived class constructor?

- The **base class's default constructor** is called automatically.
- If the base class **does not have a default constructor**, and you don't call a specific constructor explicitly, the code will **fail to compile**.

D. Given the code below - what will be the output? Why?

```
ElectricCar* eCar = new
ElectricCar("Nissan", 2022, 4, 40);
eCar->showInfo();
eCar->startEngine();
delete eCar;
```

- The parameterized constructor initializes the object with the given values.
- showInfo() is overridden in ElectricCar to display all attributes including batteryCapacity.
- startEngine() is overridden in ElectricCar and prints the custom electric engine message.
- delete cleans up the memory (assuming proper destructors are defined or default ones used safely).