Lab 04 Tasks

Task 01

The **Bookstore Management System** requires an efficient way to track and manage books. To begin, implement a **Book** class using a **default constructor** that initializes the title as "Unknown", price as 0.0, and stock as **30 units by default**. The bookstore also requires the ability to **update book details** and **track stock levels dynamically**. Additionally, a method must be implemented to **simulate book purchases**, ensuring stock levels do not go negative. If stock falls below **5 units**, the system should issue a **low-stock warning**.

Task 02

The bookstore wants more **flexibility in book management**. Modify the **Book** class to support a **parameterized constructor**, allowing the title, price, and stock to be set at object creation. To further enhance real-world application, introduce an **applyDiscount** function that **reduces the price by 5% if more than 5 copies are purchased and by 10% if more than 10 copies are purchased**. Additionally, implement a **stock validation mechanism** that ensures purchases cannot exceed available stock. If a customer attempts to purchase more than what is in stock, display an error message indicating the maximum available quantity. The system should also enforce that **book stock never drops below zero** and that all updates to price and stock are correctly reflected.

Task 03

After successfully implementing book creation and stock management, the bookstore realizes that handling multiple book objects efficiently requires **copying book details safely**. Modify the **Book** class to include a **Copy Constructor**, ensuring that when a book object is copied, it creates a deep copy rather than sharing memory references. This will prevent unwanted modifications between objects. Additionally, implement a **destructor** to release memory and display a message when an object is destroyed. In the main function, create a book object, update its details, make a copy using the copy constructor, and verify that changes to the original do not affect the copied object.

Task 04

Now that book objects can be copied safely, the bookstore wants to **dynamically calculate book prices based on additional conditions**. Modify the **Book** class to incorporate a **price calculation mechanism** that factors in **base price, discount percentage, and bulk purchase effects**. Use the **this pointer** to distinguish between local variables and class attributes when updating price-related properties. Students must ensure correct discount application while maintaining proper stock levels.

Task 05

Finally, to further streamline book management, integrate a **Member Initializer List** to ensure each book object receives a **unique Book ID** upon creation. Use a **static counter** to track book instances and assign IDs dynamically. Ensure that book IDs remain unique across all instances. Additionally, implement a method to **display all book details**, including their dynamically assigned book ID.