**Q1:**

1-Create a struct called Book that:

* Has three **public** fields: Title (string), Author (string), and Price (double).
* Has a constructor that sets all three fields.
* Has a method DisplayInfo() that prints the book details in one line.

**2-In Main:**

* Create **two Book structs** using the constructor.
* Store them in an array of Book.
* Use a foreach loop to print each book’s information

**Q2:**

1-**Create a class BankAccount**

* Public field: AccountHolder (string)
* Private field: balance (double)
* Constructor that sets AccountHolder and initial balance.
* Public method Deposit(double amount) → adds money to balance if amount > 0.
* Public method Withdraw(double amount) → subtracts money if balance is enough.
* Public method DisplayBalance() → prints the account holder name and balance.

2-**In Main method**

* Create one BankAccount object.
* Deposit money.
* Withdraw money.
* Display the balance after each operation.
* **Q 3 – Class, Encapsulation, Property, Indexer**

**Scenario:**

You are building a small library system to manage books.

**Requirements**:

1. Create Library Class with:
   * Private fields: name, address, enterFees.
   * Public properties for each field that:
     + Validate that enterFees are never negative.
   * A constructor to set up all properties.
   * A DisplayInfo() method to print the library details (like: name - address - fees)
2. Create Books Names Array inside the class too, and make:
   * A method AddBook(string bookName) to add a book to the array.
   * An indexer that allows accessing books by index and getting/setting them.
   * A method DisplayAllBooks() to show all books in the library (print all book names).
3. Test in Main:
   * Create at least 2 libraries with their data
   * Show the details of the libraries
   * Add at least 2 books to each library.
   * Access a book using the indexer
   * Display all books.

* **Q 4 – Inheritance, Polymorphism, Method Overloading, Operator Overloading, Casting Operator Overloading.**

**Scenario:**

You are building a basic BankAccount system.

There is a BankAccount base class and two derived classes: SavingsAccount and CheckingAccount.

**Requirements:**

1. Create a BankAccount class with:
   * Protected field balance.
   * Constructor to set an initial balance.
   * Method Deposit(double amount) to add to the balance.
   * Method Deposit(double amount, double bonus) to add both an amount and a bonus (method overloading).
   * Method GetBalance() to return the balance.
2. Create a SavingsAccount class inheriting from BankAccount:
   * Constructor to set an initial balance.
   * Additional method AddInterest(double rate) that increases the balance based on interest rate.
3. Create a CheckingAccount class inheriting from BankAccount:
   * Constructor to set an initial balance.
   * Additional method Withdraw(double amount) to subtract from the balance if enough funds exist.
4. Operator Overloading:
   * Overload + to merge two accounts into a new account with combined balance.
   * Overload > to compare balances of two accounts.
5. Casting Operator Overloading:
   * Add an explicit cast from BankAccount to double that returns the account’s balance.
   * Example: double balance = (double) bankAccount;
6. Test in Main:
   * Create at least one SavingsAccount and one CheckingAccount.
   * Deposit money using both versions of Deposit.
   * Compare accounts with > operator.
   * Merge accounts with + operator.
   * Cast an account to double to get its balance.

**Q 5:**

a) Create a base class BaseClass with a virtual method DisplayMessage that prints "Message from BaseClass".

b) Create a derived class DerivedClass1 that overrides the DisplayMessage method using the override keyword.

C) Create another derived class DerivedClass2 that hides the DisplayMessage method using the new keyword.

Then explain the difference between using override and new (using binding behavior

**Q 6:** 1-Define Class Duration To include Three Attributes Hours, Minutes and Seconds.

2-Override All System. Object Members [To String(), Equals(),GetHashCode() ] .

3-Define All Required Constructors to Produce this output:

Duration D1 =new Duration (1,10,15);

D1.ToString();

Output: Hours: 1, Minutes :10, Seconds :15

**Q7:**

Define Class Maths that has four methods:

Add()

Subtract()

Multiply()

Divide()

Each of them takes two parameters. Call each method in Main ().

NOTE : Modify the program so that you do not have to create an instance of class to call the four methods.

**Q8:**

You are tasked with designing a system for an e-commerce platform that calculates discounts for different types of users and products.

This system should utilize abstraction and include the following parts:

Part 1: Abstract Discount Class

Create an abstract class Discount with:

An abstract method CalculateDiscount(decimal price, int quantity) that returns the discount amount based on the original price and quantity.

A Name property to store the type of discount.

Part 2: Specific Discounts

Implement the following concrete discount classes:

PercentageDiscount:

Accepts a percentage (e.g., 10%).

Formula: Discount Amount=Price×Quantity×(Percentage/100)

FlatDiscount:

Accepts a fixed amount to be deducted (e.g., $50).

Formula: Discount Amount=Flat Amount×min(Quantity,1)

BuyOneGetOneDiscount:

Applies a 50% discount if the quantity is greater than 1.

Formula: Discount Amount=(Price/2)×(Quantity÷2)

**Bonus**

complete Third Project by this Part 3: Discount Applicability

Create an abstract class User with:

A Name property to store the user name.

An abstract method GetDiscount() that returns a Discount object.

Implement the following specific user types:

RegularUser: Applies a PercentageDiscount of 5%.

PremiumUser: Applies a FlatDiscount of $100.

GuestUser: No discount is applied

Part 4: Integration

Write a program that:

Ask the user to input their type (Regular, Premium, or Guest).

Allows the user to input product details (price and quantity).

Calculates and displays the total discount and final price after applying the appropriate discount.