# Problem 1) Sum of Natural Numbers (Recursion)

Create a recursive method SumNumbers(int n) that calculates the sum of all natural numbers from 1 to n.  
**Example:**  
Console.WriteLine(SumNumbers(5));  
Expected Output: 15

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 2) Product of Natural Numbers (Recursion)

Create a recursive method ProductNumbers(int n) that calculates the product of all natural numbers from 1 to n.  
**Example:**  
Console.WriteLine(ProductNumbers(5));  
Expected Output: 120

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Problem 3) Optional & Named Parameters – Product Order

Create a method PlaceOrder(string productName, int quantity = 1, string customer = "Guest", bool expressDelivery = false)  
that prints order details. Call this method in 3 different ways using:  
1. Only required parameter  
2. Required + one optional parameter  
3. Using named parameters in a different order  
**Expected Output:**  
Product: Laptop | Quantity: 1 | Customer: Guest | Express Delivery: False  
Product: Keyboard | Quantity: 5 | Customer: Guest | Express Delivery: False  
Product: Monitor | Quantity: 2 | Customer: Alice | Express Delivery: True

## Problem 4) Discounted Order

Modify PlaceOrder method to include an optional parameter double discount = 0.  
If discount > 0, print the discounted total price. (Assume each product costs $100 for simplicity)  
**Example:**  
PlaceOrder("Tablet", 2, "Bob", true, 0.1);  
Expected Output:  
Product: Tablet | Quantity: 2 | Customer: Bob | Express Delivery: True | Discount: 10% | Total Price: $180

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Problem 5) Extension Method – Reverse String

Create an extension method ReverseText(this string text) that returns the string in reverse order.  
If the string is empty or null, return an empty string.  
Example:  
string message = "hello";  
Console.WriteLine(message.ReverseText());  
Expected Output: olleh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Problem 6) Extension Method – Character Count

Create an extension method CountCharacters(this string text) that returns the number of characters in the string.  
If the string is empty or null, return 0.  
Example:  
string message = "Hello";  
Console.WriteLine(message.CountCharacters());  
Expected Output: 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Problem 7) Extension Method – IsEven

Create an extension method for int called IsEven that returns true if the number is even and false otherwise.  
**Example:**  
int x = 8;  
Console.WriteLine(x.IsEven());  
Expected Output: True

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Problem 8) Extension Method – IsPrime

Create an extension method for int called IsPrime that returns true if the number is prime and false otherwise.  
**Example:**  
int x = 7;  
Console.WriteLine(x.IsPrime());  
Expected Output: True

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Problem 9) Records

Create a record called Product with the following properties:  
- Id (int)  
- Name (string)  
- Price (double)  
  
Create two instances of Product and compare them using the equality operator (==) to check if they are equal.  
**Example:**  
Product p1 = new Product(1, "Laptop", 1500);  
Product p2 = new Product(1, "Laptop", 1500);  
Console.WriteLine(p1 == p2);  
Expected Output: True

## Problem 10) Record with Deconstruction

Modify the Product record to support deconstruction and print its properties individually.  
**Example:**  
var (id, name, price) = p1;  
Console.WriteLine($"Id: {id}, Name: {name}, Price: {price}");  
Expected Output:  
Id: 1, Name: Laptop, Price: 1500