Name-Sangram Mandal

Superset Id-6363848

Week-1(Handson-Exercise)

1. Data Structure & Algorithms :-

**Exercise 2: E-commerce Platform Search Function**

**Code :-**

**Product.cs :-**

public class Product

{

    public int ProductId { get; set; }

    public string ProductName { get; set; }

    public string Category { get; set; }

    public Product(int id, string name, string category)

    {

        ProductId = id;

        ProductName = name;

        Category = category;

    }

}

**Program.cs :-**

using System;

using System.Linq;

class Program

{

    static void Main(string[] args)

    {

        Product[] productList = new Product[]

        {

            new Product(101, "Laptop", "Electronics"),

            new Product(102, "Shoes", "Fashion"),

            new Product(103, "Camera", "Electronics"),

            new Product(104, "Watch", "Accessories"),

            new Product(105, "Tablet", "Electronics")

        };

        Console.Write("Enter product name to search: ");

        string target = Console.ReadLine();

        Product linearResult = LinearSearch(productList, target);

        Console.WriteLine(linearResult != null

            ? $"Linear Search: Found {linearResult.ProductName} in {linearResult.Category}"

            : "Linear Search: Product not found");

        Product[] sortedProducts = productList.OrderBy(p => p.ProductName).ToArray();

        Product binaryResult = BinarySearch(sortedProducts, target);

        Console.WriteLine(binaryResult != null

            ? $"Binary Search: Found {binaryResult.ProductName} in {binaryResult.Category}"

            : "Binary Search: Product not found");

        Console.WriteLine("\nTime Complexity:");

        Console.WriteLine("Linear Search: O(n) - good for small or unsorted data.");

        Console.WriteLine("Binary Search: O(log n) - faster but requires sorted data.");

    }

//Linear search

    static Product LinearSearch(Product[] products, string targetName)

    {

        foreach (Product product in products)

        {

            if (product.ProductName.Equals(targetName, StringComparison.OrdinalIgnoreCase))

                return product;

        }

        return null;

    }

    //binary search

    static Product BinarySearch(Product[] products, string targetName)

    {

        int left = 0;

        int right = products.Length - 1;

        while (left <= right)

        {

            int mid = left + (right - left) / 2;

            int comparison = string.Compare(products[mid].ProductName, targetName, StringComparison.OrdinalIgnoreCase);

            if (comparison == 0)

                return products[mid];

            else if (comparison < 0)

                left = mid + 1;

            else

                right = mid - 1;

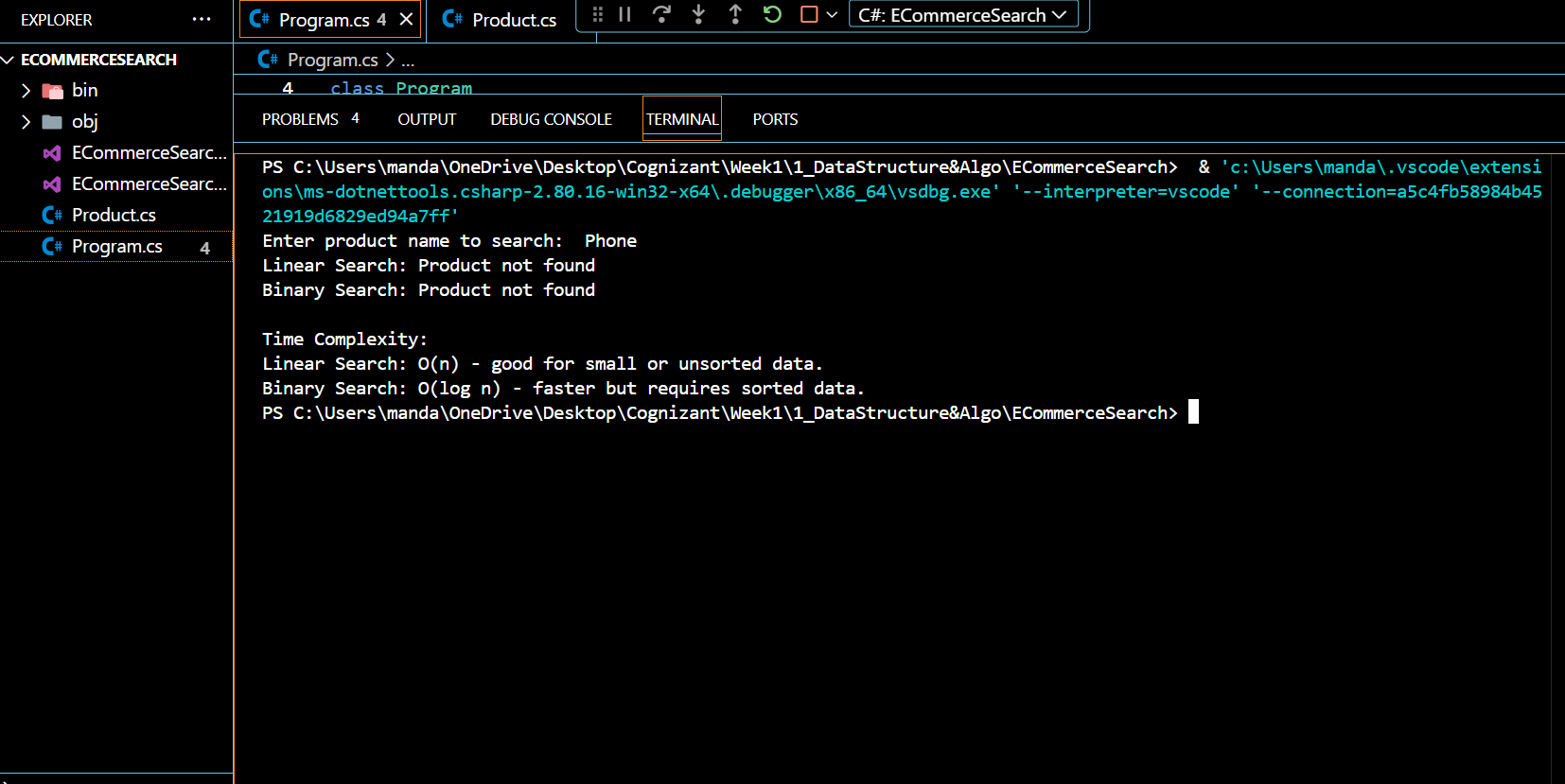
        }

        return null;

    }

}

**Output :-**

****

**Exercise 7: Financial Forecasting**

**Code :-**

**Program.cs :-**

using System;

class Program

{

    static void Main(string[] args)

    {

        Console.Write("Enter current value (e.g., 1000): ");

        double currentValue = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter annual growth rate (in %, e.g., 5): ");

        double growthRate = Convert.ToDouble(Console.ReadLine()) / 100;

        Console.Write("Enter number of years to forecast: ");

        int years = Convert.ToInt32(Console.ReadLine());

        double futureValue = CalculateFutureValueRecursive(currentValue, growthRate, years);

        Console.WriteLine($"\nForecasted value after {years} years: {futureValue:F2}");

    }

    static double CalculateFutureValueRecursive(double currentValue, double growthRate, int years)

    {

        if (years == 0)

            return currentValue;

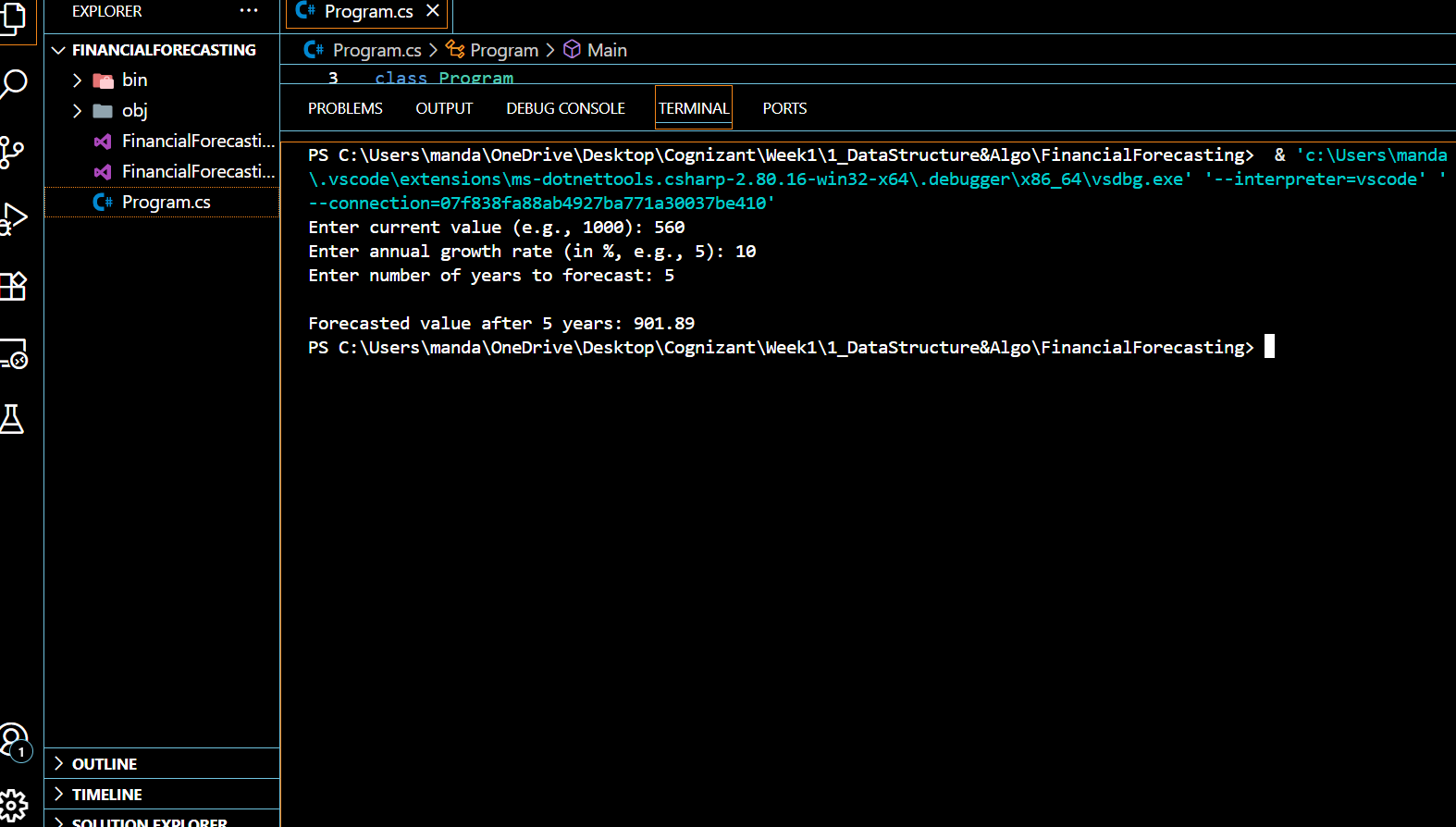
        return CalculateFutureValueRecursive(currentValue, growthRate, years - 1) \* (1 + growthRate);

    }

}

**Output :-**

**For current value as 560 ,annual growth rate as 10 percent anf number of year as 5 output will be :-**

****