# **Name- Sayandeep Dey (SupersetID:** **6363427)**

# **WEEK – 1 (Handson- Exercises)**

1. **Data structures and Algorithms:**

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

**Code:**

**In 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;

    }

    public override string ToString()

    {

        return $"ID: {ProductId}, Name: {ProductName}, Category: {Category}";

    }

}

**In SearchEngine.cs:**using System;

public class SearchEngine

{

    // Linear Search

    public static Product? LinearSearch(Product[] products, string target)

    {

        foreach (var product in products)

        {

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

                return product;

        }

        return null;

    }

    // Binary Search (sorted by ProductName)

    public static Product? BinarySearch(Product[] sortedProducts, string target)

    {

        int left = 0;

        int right = sortedProducts.Length - 1;

        while (left <= right)

        {

            int mid = (left + right) / 2;

            int cmp = string.Compare(sortedProducts[mid].ProductName, target, StringComparison.OrdinalIgnoreCase);

            if (cmp == 0)

                return sortedProducts[mid];

            else if (cmp < 0)

                left = mid + 1;

            else

                right = mid - 1;

        }

        return null;

    }

}

**In Program.cs:**

using System;

using System.Linq;

class Program

{

    static void Main()

    {

        Product[] products = new Product[]

        {

            new Product(1, "iPhone", "Electronics"),

            new Product(2, "Shoes", "Footwear"),

            new Product(3, "Laptop", "Electronics"),

            new Product(4, "Book", "Education"),

            new Product(5, "T-Shirt", "Clothing")

        };

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

        string? input = Console.ReadLine();

        if (string.IsNullOrWhiteSpace(input))

        {

            Console.WriteLine("Invalid product name.");

            return;

        }

        Console.WriteLine("\n Linear Search Result:");

        var linearResult = SearchEngine.LinearSearch(products, input);

        Console.WriteLine(linearResult != null ? linearResult : "Product not found");

        Console.WriteLine("\n Binary Search Result:");

        var sorted = products.OrderBy(p => p.ProductName).ToArray();

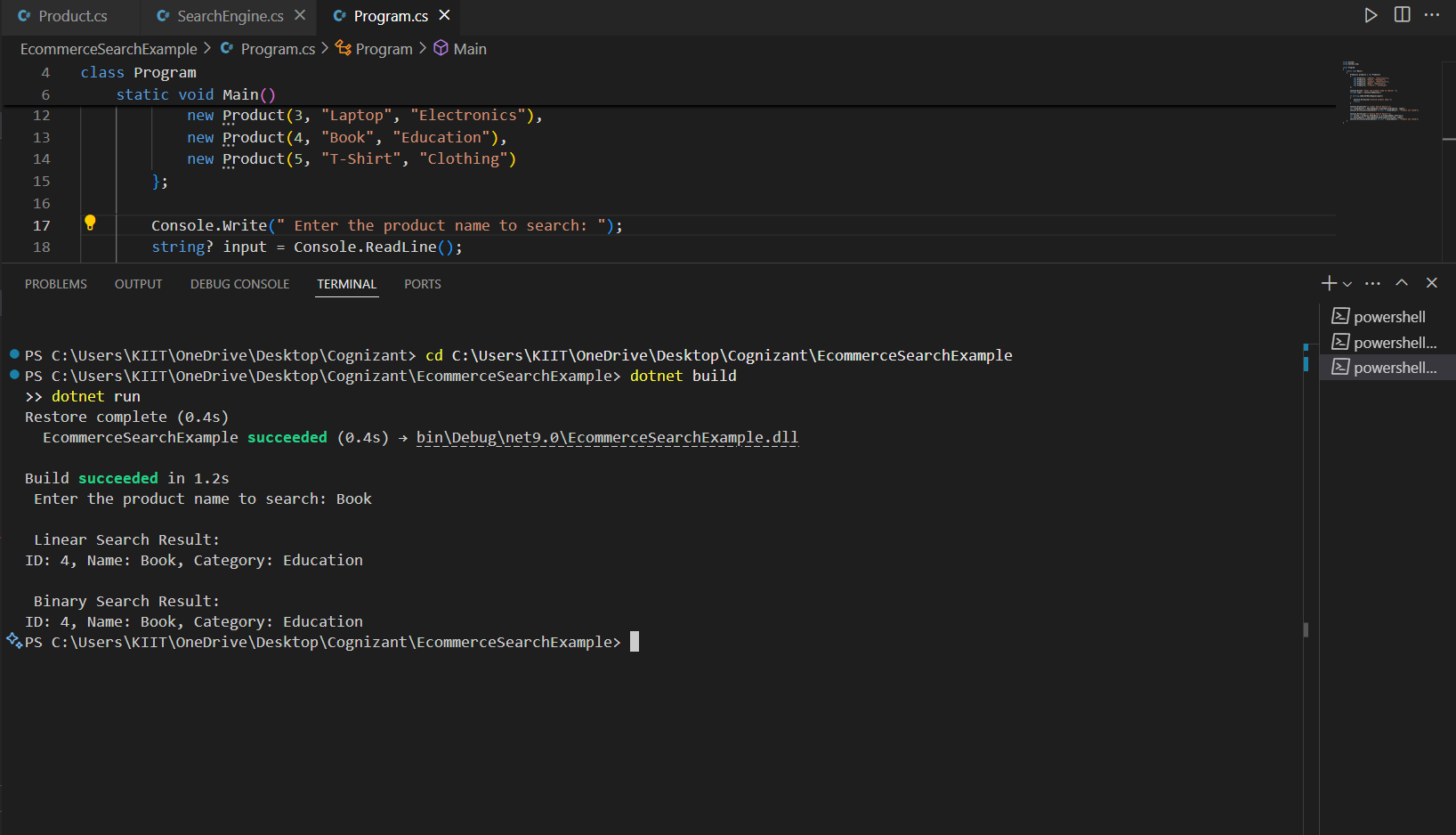
        var binaryResult = SearchEngine.BinarySearch(sorted, input);

        Console.WriteLine(binaryResult != null ? binaryResult : "Product not found");

    }

}

**Output:**

**  
Analysis:**

| **Algorithm** | **Time Complexity** | **Space Complexity** | **Notes** |
| --- | --- | --- | --- |
| **Linear Search** | **O(n)** | **O(1)** | **Works with unsorted data** |
| **Binary Search** | **O(log n)** | **O(1)** | **Requires sorted data** |

**Exercise 7: Financial Forecasting:**

**Code:**

**In FinancialForecasting.cs:**

using System;

public class FinancialForecast

{

    public static double PredictFutureValue(double initialValue, double growthRate, int years)

    {

        if (years == 0)

            return initialValue;

        return PredictFutureValue(initialValue, growthRate, years - 1) \* (1 + growthRate);

    }

}

**In Program.cs:**

using System;

class Program

{

    static void Main()

    {

        Console.WriteLine(" Financial Forecasting Tool");

        // User input for initial value

        Console.Write("Enter the initial amount (e.g., 1000): ");

        if (!double.TryParse(Console.ReadLine(), out double initialValue) || initialValue < 0)

        {

            Console.WriteLine("Invalid initial value.");

            return;

        }

        // User input for growth rate

        Console.Write("Enter the annual growth rate in % (e.g., 10 for 10%): ");

        if (!double.TryParse(Console.ReadLine(), out double growthRate) || growthRate < 0)

        {

            Console.WriteLine("Invalid growth rate.");

            return;

        }

        // Convert percentage to decimal

        growthRate = growthRate / 100.0;

        // User input for number of years

        Console.Write("Enter the number of years to forecast (e.g., 5): ");

        if (!int.TryParse(Console.ReadLine(), out int years) || years < 0)

        {

            Console.WriteLine("Invalid number of years.");

            return;

        }

        // Call recursive method

        double futureValue = PredictFutureValue(initialValue, growthRate, years);

        // Display result

        Console.WriteLine($"\n Future value after {years} years: {futureValue:C2}");

    }

    // Recursive method to calculate future value

    static double PredictFutureValue(double initialValue, double growthRate, int years)

    {

        if (years == 0)

            return initialValue;

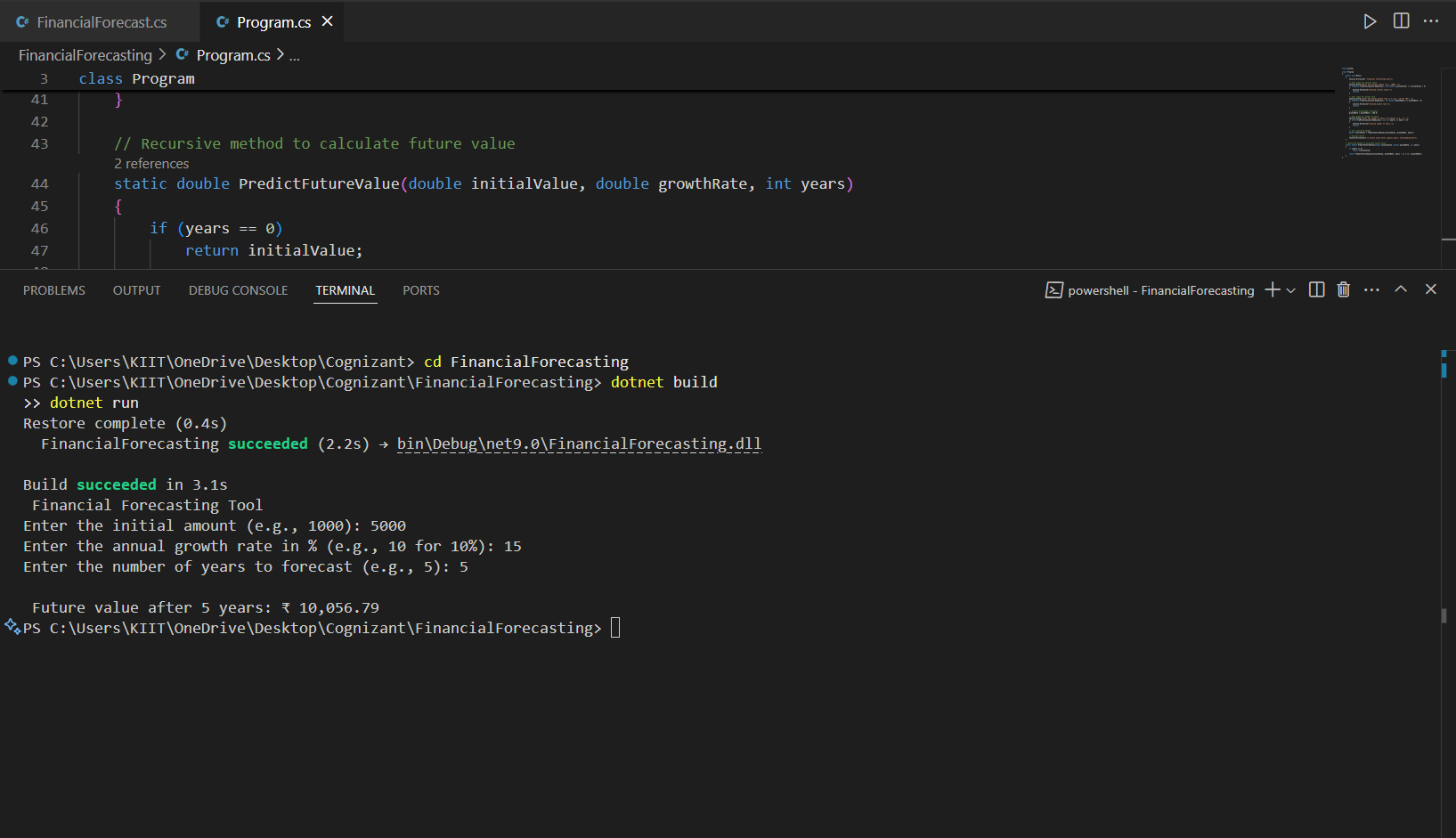
        return PredictFutureValue(initialValue, growthRate, years - 1) \* (1 + growthRate);

    }

}

**Output:**

**For the initial amount of 5000 and the annual growth rate of 15% and for 5 years the output will be:**

****