Algorithms\_Data Structures

Exercise 2: E-commerce Platform Search Function

Code:

**//File: Product.cs**

public class Product

{

    public int productId;

    public string productName;

    public string category;

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

    {

        productId = id;

        productName = name;

        category = cat;

    }

}

**//File: SearchAlgo.cs**

using System;

public class SearchAlgo

{

    // Linear Search

    public static int LinearSearch(Product[] products, string name)

    {

        for (int i = 0; i < products.Length; i++)

        {

            if (products[i].productName == name)

            {

                return i;

            }

        }

        return -1;

    }

    // Bubble Sort

    public static void SortByName(Product[] products)

    {

        for (int i = 0; i < products.Length - 1; i++)

        {

            for (int j = 0; j < products.Length - i - 1; j++)

            {

                if (string.Compare(products[j].productName, products[j + 1].productName) > 0)

                {

                    Product temp = products[j];

                    products[j] = products[j + 1];

                    products[j + 1] = temp;

                }

            }

        }

    }

    // Binary Search

    public static int BinarySearch(Product[] products, string name)

    {

        int low = 0;

        int high = products.Length - 1;

        while (low <= high)

        {

            int mid = (low + high) / 2;

            int result = string.Compare(products[mid].productName, name);

            if (result == 0)

                return mid;

            else if (result < 0)

                low = mid + 1;

            else

                high = mid - 1;

        }

        return -1;

    }

}

**//File: Program.cs**

using System;

class Program

{

static void Main(string[] args)

{

Product[] productList = new Product[5];

productList[0] = new Product(1, "Laptop", "Electronics");

productList[1] = new Product(2, "Shirt", "Clothing");

productList[2] = new Product(3, "Book", "Stationery");

productList[3] = new Product(4, "Phone", "Electronics");

productList[4] = new Product(5, "Shoes", "Footwear");

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

string userSearchInput = Console.ReadLine();

// Linear Search

int indexLinear = SearchAlgorithms.LinearSearch(productList, userSearchInput);

Console.WriteLine(indexLinear != -1

? $"Linear Search: Found at index {indexLinear}"

: "Linear Search: Not found");

// Binary Search

SearchAlgorithms.SortByName(productList);

int indexBinary = SearchAlgorithms.BinarySearch(productList, userSearchInput);

Console.WriteLine(indexBinary != -1

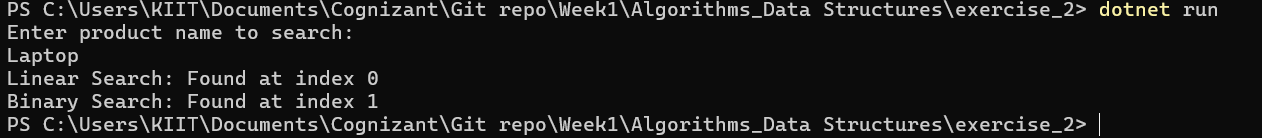
? $"Binary Search: Found at index {indexBinary}"

: "Binary Search: Not found");

}

}

Output:



Exercise 7: Financial Forecasting

Code:

**//File: Forecaster.cs**

using System;

public class Forecaster

{

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

    {

        if (years == 0)

        {

            return currentValue; // base case

        }

        else

        {

            double nextYearValue = currentValue \* (1 + growthRate);

            return PredictFutureValue(nextYearValue, growthRate, years - 1);

        }

    }

}

**//File: Program.cs**

using System;

class Program{

    static void Main(string[] args){

        Console.WriteLine("Enter current value (e.g., 10000):");

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

        Console.WriteLine("Enter annual growth rate (e.g., 0.10 for 10%):");

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

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

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

        double futureValue = Forecaster.PredictFutureValue(currentValue, growthRate, years);

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

    }

}

Output:

