**Data structures and Algorithms**

**Exercise 1: E-commerce Platform Search Function**

**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}";**

**}**

**}**

**Search.cs :-**

**using System;**

**public static class Search**

**{**

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

**{**

**foreach (var product in products)**

**{**

**if (product.ProductName.Equals(name, StringComparison.OrdinalIgnoreCase))**

**return product;**

**}**

**return null;**

**}**

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

**{**

**int left = 0;**

**int right = products.Length - 1;**

**while (left <= right)**

**{**

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

**int comparison = string.Compare(products[mid].ProductName, name, true);**

**if (comparison == 0)**

**return products[mid];**

**else if (comparison < 0)**

**left = mid + 1;**

**else**

**right = mid - 1;**

**}**

**return null;**

**}**

**}**

**Program.cs :-**

**using System;**

**class Program**

**{**

**static void Main()**

**{**

**Product[] products = new Product[]**

**{**

**new Product(1, "Laptop", "Electronics"),**

**new Product(2, "Shirt", "Clothing"),**

**new Product(3, "Phone", "Electronics"),**

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

**}**

**Console.WriteLine(" Linear Search:");**

**var linearResult = SearchDemo.LinearSearch(products, "Laptop");**

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

**Array.Sort(products, (x, y) => x.ProductName.CompareTo(y.ProductName));**

**Console.WriteLine("\nBinary Search:");**

**var binaryResult = SearchDemo.BinarySearch(products, "Shirt");**

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

**Console.ReadKey();**

**}**

**}**

**OUTPUT:-**



**Exercise 2: Financial Forecasting**

**Forecaster.cs :-**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace FinancialForecasting**

**{**

**internal class Forecaster**

**{**

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

**{**

**if (years == 0)**

**return currentValue;**

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

**}**

**}**

**}**

**Program.cs :-**

**using System;**

**namespace FinancialForecasting**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Financial Forecasting Tool");**

**Console.Write("Enter current value: ");**

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

**Console.Write("Enter annual growth rate (e.g., 0.05 for 5%): ");**

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

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

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

**Forecaster forecaster = new Forecaster();**

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

**Console.WriteLine($"\n Future Value after {years} years: {futureValue:F2}");**

**Console.ReadKey();**

**}**

**}**

**}**

**OUTPUT :-**

****