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

using System;

class Product {

    public string ProductId;

    public string ProductName;

    public string Category;

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

        ProductId = id;

        ProductName = name;

        Category = category;

    }

}

class ProductSearch {

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

        foreach (var p in products)

            if (p.ProductName.Equals(name, StringComparison.OrdinalIgnoreCase))

                return p;

        return null;

    }

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

        Array.Sort(products, (a, b) => a.ProductName.CompareTo(b.ProductName));

        int low = 0, high = products.Length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (cmp == 0) return products[mid];

            if (cmp < 0) low = mid + 1;

            else high = mid - 1;

        }

        return null;

    }

    public static void Main() {

        Product[] products = {

            new Product("1", "Apple", "Fruits"),

            new Product("2", "Banana", "Fruits"),

            new Product("3", "Carrot", "Vegetables"),

            new Product("4", "Dates", "Dry Fruits")

        };

        var result1 = LinearSearch(products, "Carrot");

        Console.WriteLine("Linear Search: " + (result1 != null ? result1.ProductName : "Not Found"));

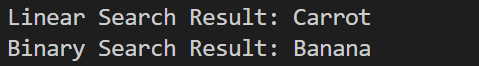
        var result2 = BinarySearch(products, "Banana");

        Console.WriteLine("Binary Search: " + (result2 != null ? result2.ProductName : "Not Found"));

    }

}

**Output:-**

****

**Exercise 7: Financial Forecasting**

using System;

class FinancialForecast {

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

        if (years == 0) return currentValue;

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

    }

    public static double PredictFutureMemo(double currentValue, double growthRate, int years, double[] memo) {

        if (years == 0) return currentValue;

        if (memo[years] != 0) return memo[years];

        memo[years] = PredictFutureMemo(currentValue \* (1 + growthRate), growthRate, years - 1, memo);

        return memo[years];

    }

    public static void Main() {

        double currentValue = 10000;

        double growthRate = 0.05; // 5%

        int years = 5;

        double result = PredictFutureValue(currentValue, growthRate, years);

        Console.WriteLine("Future Value (recursive): " + result);

        double[] memo = new double[years + 1];

        double resultMemo = PredictFutureMemo(currentValue, growthRate, years, memo);

        Console.WriteLine("Future Value (memoized): " + resultMemo);

    }

}

**Output:-**

