**Superset Id- 6363535**

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

**Code:**

[**Product.cs**](http://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**](http://program.cs)

using System;

using System.Linq

namespace EcommerceSearchFunction

{

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;

}

}

class Program

{

static void Main(string[] args)

}

Product[] products = {

new Product(1, "Tooth brush", "pen"),

new Product(2, "Shirt", "umbrella"),

new Product(3, "Smartwatch", "Bag"),

new Product(4, "Paint", "Stationery"),

new Product(5, "Shoes", "flower pot")

};

Console.WriteLine(" Linear Search:");

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

Console.WriteLine(result1 != null ? $"Found: {result1.ProductName}" : "Not Found");

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

Console.WriteLine(" Binary Search:");

var result2 = BinarySearch(sortedProducts, "Paint", 0, sortedProducts.Length - 1);

Console.WriteLine(result2 != null ? $"Found: {result2.ProductName}" : "Not Found");

}

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[] sortedProducts, string name, int left, int right)

{

if (left <= right)

{

int mid = (left + right) / 2;

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

if (comparison == 0)

return sortedProducts[mid];

else if (comparison < 0)

return BinarySearch(sortedProducts, name, left, mid - 1);

else

return BinarySearch(sortedProducts, name, mid + 1, right);

}

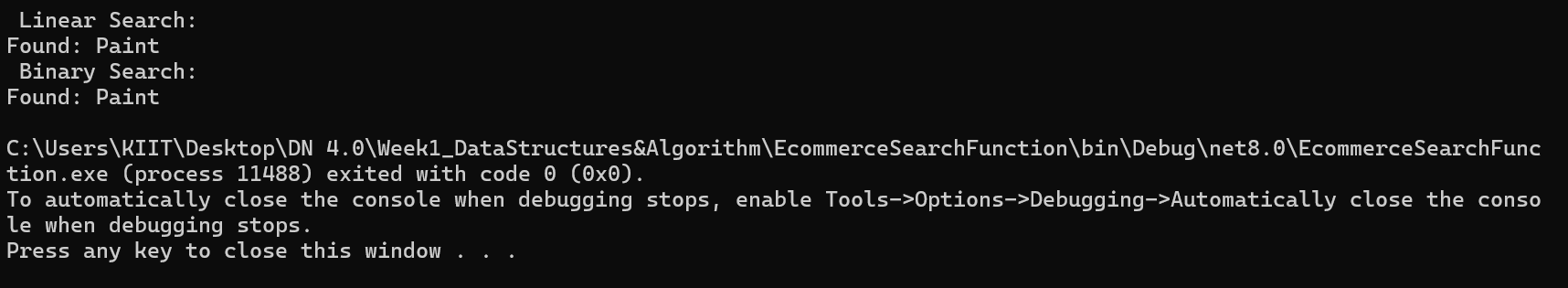
return null;

}

}

}

**Output:**

****

**Exercise 7: Financial Forecasting**

[**Program.cs**](http://program.cs)

using System;

namespace FinancialForecasting

{

class Program

{

static void Main(string[] args)

{

double presentValue = 10000;

double annualRate = 0.05;

int years = 5;

double futureValueRecursive = ForecastRecursive(presentValue, annualRate, years);

Console.WriteLine($"[Recursive] Future value after {years} years: {futureValueRecursive:F2}");

double futureValueIterative = ForecastIterative(presentValue, annualRate, years);

Console.WriteLine($"[Iterative] Future value after {years} years: {futureValueIterative:F2}");

}

public static double ForecastRecursive(double presentValue, double rate, int years)

{

if (years == 0)

return presentValue;

return ForecastRecursive(presentValue, rate, years - 1) \* (1 + rate);

}

public static double ForecastIterative(double presentValue, double rate, int years)

{

for (int i = 0; i < years; i++)

{

presentValue \*= (1 + rate);

}

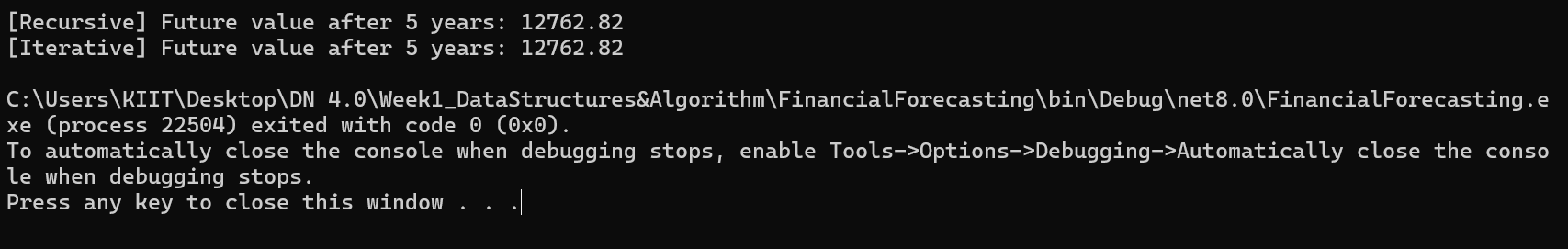
return presentValue;

}

}

}

**Output:**

****