**WEEK1: E-commerce Platform Search Function**

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Program.cs**

**using System;**

**public class Product**

**{**

**public int productID;**

**public string productName;**

**public string category;**

**public Product(int id,string name,string cate)**

**{**

**productID = id;**

**productName = name;**

**category = cate;**

**}**

**}**

**class Program**

**{**

**static void Main()**

**{**

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

**int n = int.Parse(Console.ReadLine());**

**Product[] product\_arr = new Product[n];**

**for (int i = 0; i < n; i++)**

**{**

**Console.WriteLine($"\nEnter details for Product {i + 1}:");**

**Console.Write("Product ID: ");**

**int id = int.Parse(Console.ReadLine());**

**Console.Write("Product Name: ");**

**string name = Console.ReadLine();**

**Console.Write("Category: ");**

**string category = Console.ReadLine();**

**product\_arr[i] = new Product(id, name, category);**

**}**

**Array.Sort(product\_arr, (p1, p2) => string.Compare(p1.productName, p2.productName, StringComparison.OrdinalIgnoreCase));**

**Console.Write("\nEnter product name to search: ");**

**string searchName = Console.ReadLine();**

**int linearIndex = LinearSearch(product\_arr, searchName);**

**Console.WriteLine("Linear Search: " + (linearIndex == -1 ? "Product not found" : $"Found at index {linearIndex}"));**

**int binaryIndex = BinarySearch(product\_arr, searchName);**

**Console.WriteLine("Binary Search: " + (binaryIndex == -1 ? "Product not found" : $"Found at index {binaryIndex}"));**

**}**

**public static int LinearSearch(Product[] product\_arr, string targetName)**

**{**

**for (int i = 0; i < product\_arr.Length; i++)**

**{**

**if (product\_arr[i].productName.Equals(targetName, StringComparison.OrdinalIgnoreCase))**

**{**

**return i;**

**}**

**}**

**return -1;**

**}**

**public static int BinarySearch(Product[] product\_arr, string targetName)**

**{**

**int left = 0;**

**int right = product\_arr.Length - 1;**

**while (left <= right)**

**{**

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

**int comp = string.Compare(product\_arr[mid].productName, targetName, StringComparison.OrdinalIgnoreCase);**

**if (comp == 0)**

**return mid;**

**else if (comp < 0)**

**left = mid + 1;**

**else**

**right = mid - 1;**

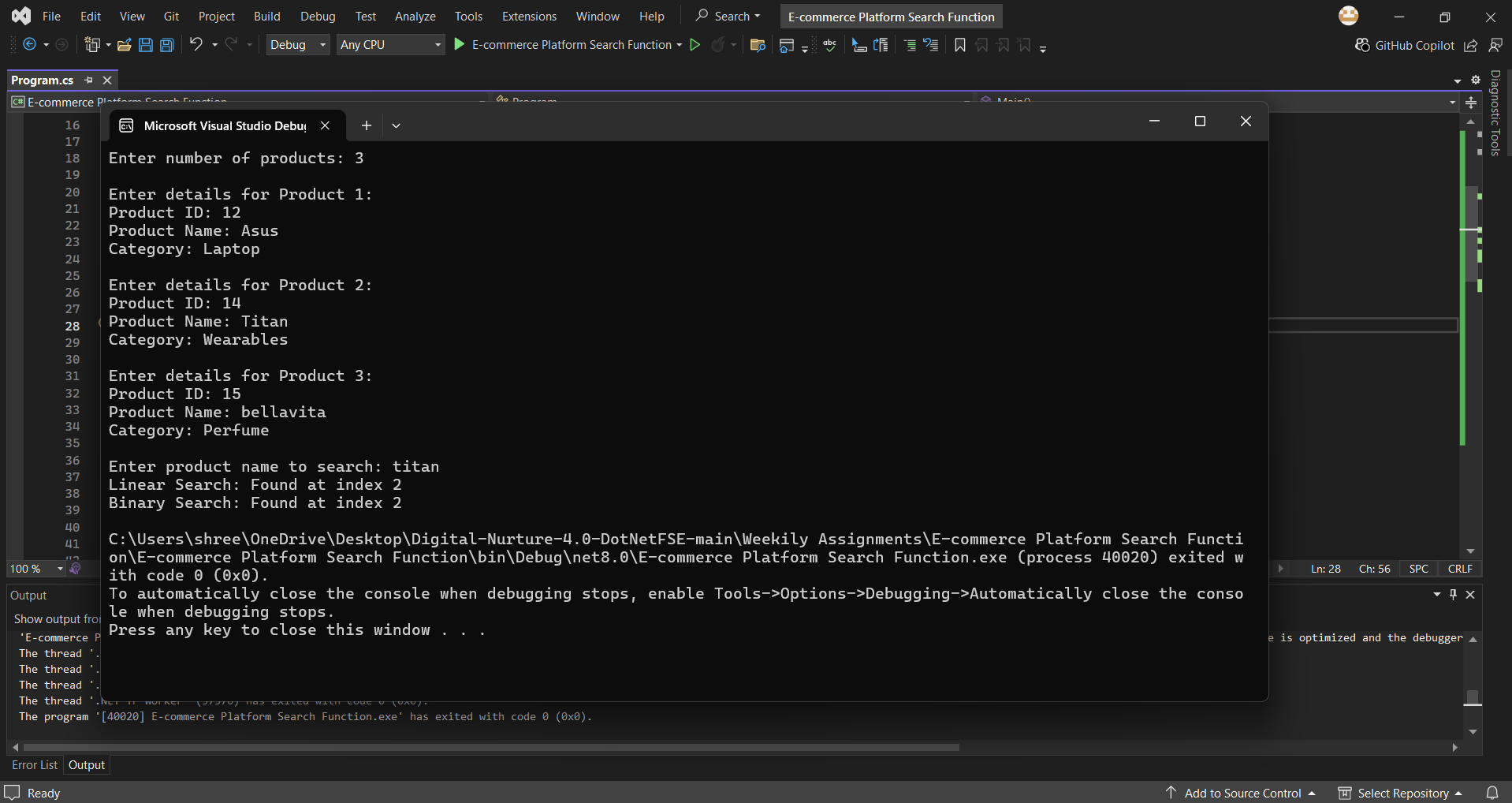
**}**

**return -1;**

**}**

**}**

**OUTPUT:**

****

**Analysis:**

Compare the time complexity of linear and binary search algorithms.

**1.Linear Search**

**Logic:**

* Check each element one by one and Works on unsorted or sorted data.

|  |  |  |
| --- | --- | --- |
| Case | Explanation | Time Complexity |
| Best Case | Element is at the beginning | O(1) |
| Average Case | Element is somewhere in the middle | O(n) |
| Worst Case | Element is last or not present | O(n) |

2. **Binary Search**

**Logic:**

* Repeatedly divide the sorted array in half and compare with the middle element and Works only on sorted data.

|  |  |  |
| --- | --- | --- |
| Case | Explanation | Time Complexity |
| Best Case | Element is at the middle | O(1) |
| Average Case | Divide and check recursively | O(log n) |
| Worst Case | Takes log n steps to find or not find | O(log n) |

Discuss which algorithm is more suitable for your platform and why.

Answer:

**Binary Search** is more suitable for an e-commerce platform. Since product lists can be sorted and fast search is important, binary search is the better choice for an e-commerce platform.

* It is **faster** than linear search.
* It works well when the product list is **sorted**.
* E-commerce platforms have **many products**, so speed is important.
* Binary search helps users get **quick search results**.

**Linear Search :**

* Slower when there are many products.
* Checks each item one by one.
* Good only for small or unsorted lists.