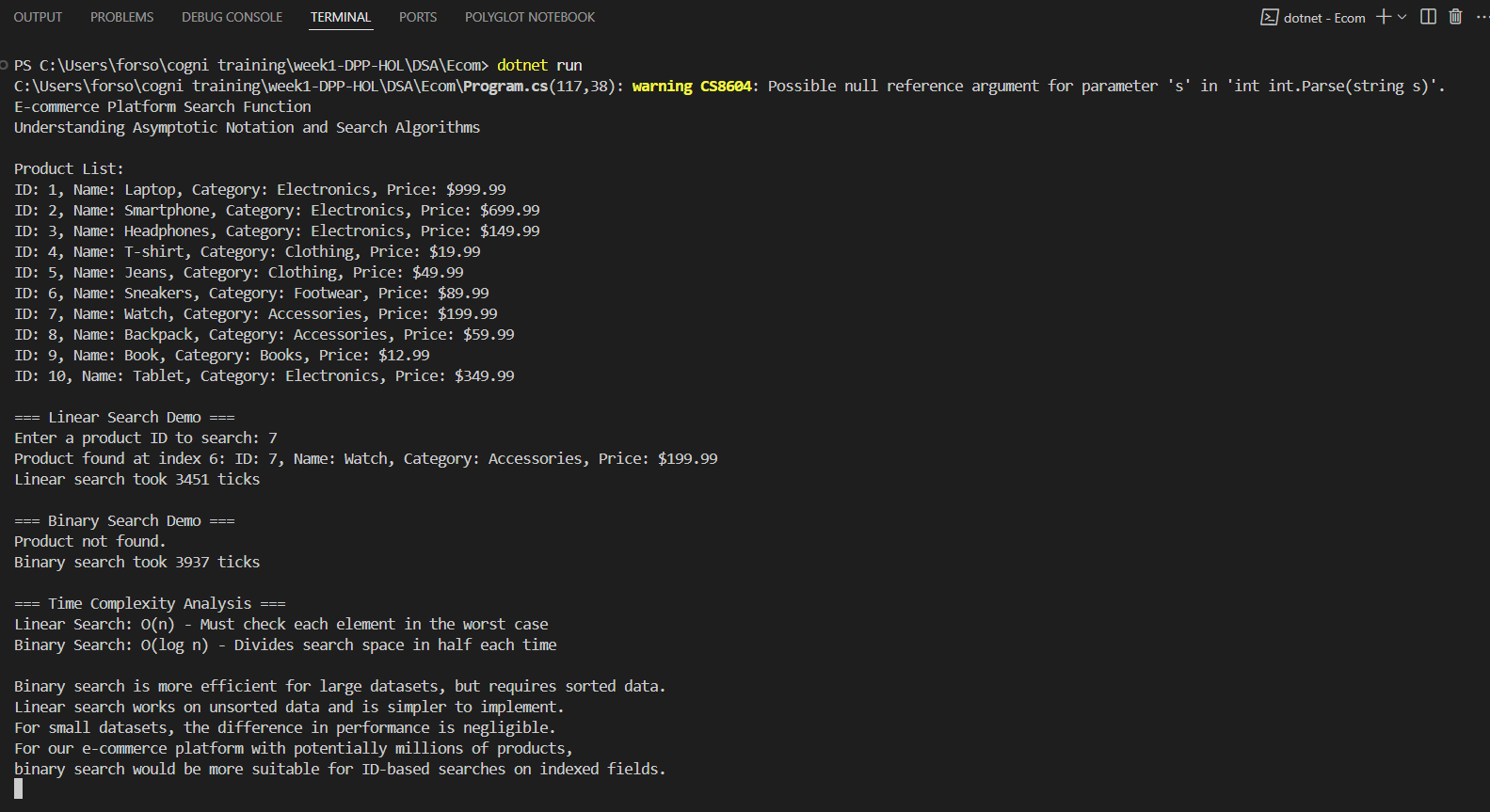
**Exercise 2: E-commerce Platform Search Function**

**Scenario:**

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

**OUTPUT:**



**CODE:**

// Exercise 3: E-commerce Platform Search Function

// Search Algorithms Implementation

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace EcommerceSearchFunction

{

    // Product class with attributes for searching

    public class Product

    {

        public int ProductId { get; set; }

        public string ProductName { get; set; }

        public string Category { get; set; }

        public decimal Price { get; set; }

        public Product(int id, string name, string category, decimal price)

        {

            ProductId = id;

            ProductName = name;

            Category = category;

            Price = price;

        }

        public override string ToString()

        {

            return $"ID: {ProductId}, Name: {ProductName}, Category: {Category}, Price: ${Price}";

        }

    }

    public class SearchAlgorithms

    {

        // Linear search implementation

        public static int LinearSearch<T>(T[] array, Predicate<T> match)

        {

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

            {

                if (match(array[i]))

                {

                    return i;

                }

            }

            return -1; // Not found

        }

        // Binary search implementation (requires sorted array)

        public static int BinarySearch<T>(T[] sortedArray, Predicate<T> match, Func<T, T, int> compare)

        {

            int left = 0;

            int right = sortedArray.Length - 1;

            while (left <= right)

            {

                int mid = (left + right) / 2;

                if (match(sortedArray[mid]))

                {

                    return mid;

                }

                // Assuming sortedArray[0] is the smallest value

                T midValue = sortedArray[mid];

                if (compare(midValue, sortedArray[0]) < 0)

                {

                    right = mid - 1;

                }

                else

                {

                    left = mid + 1;

                }

            }

            return -1; // Not found

        }

    }

    public class Program

    {

        static void Main(string[] args)

        {

            Console.WriteLine("E-commerce Platform Search Function");

            Console.WriteLine("Understanding Asymptotic Notation and Search Algorithms");

            // Create a list of products

            List<Product> products = new List<Product>

            {

                new Product(1, "Laptop", "Electronics", 999.99m),

                new Product(2, "Smartphone", "Electronics", 699.99m),

                new Product(3, "Headphones", "Electronics", 149.99m),

                new Product(4, "T-shirt", "Clothing", 19.99m),

                new Product(5, "Jeans", "Clothing", 49.99m),

                new Product(6, "Sneakers", "Footwear", 89.99m),

                new Product(7, "Watch", "Accessories", 199.99m),

                new Product(8, "Backpack", "Accessories", 59.99m),

                new Product(9, "Book", "Books", 12.99m),

                new Product(10, "Tablet", "Electronics", 349.99m)

            };

            // Convert to array for search algorithms

            Product[] productArray = products.ToArray();

            // Sorted array by ProductId for binary search

            Product[] sortedByIdArray = new Product[productArray.Length];

            Array.Copy(productArray, sortedByIdArray, productArray.Length);

            Array.Sort(sortedByIdArray, (p1, p2) => p1.ProductId.CompareTo(p2.ProductId));

            Console.WriteLine("\nProduct List:");

            foreach (var product in productArray)

            {

                Console.WriteLine(product);

            }

            // Demo: Linear Search

            Console.WriteLine("\n=== Linear Search Demo ===");

            Console.Write("Enter a product ID to search: ");

            int searchId = int.Parse(Console.ReadLine());

            Stopwatch stopwatch = new Stopwatch();

            stopwatch.Start();

            int linearResult = SearchAlgorithms.LinearSearch(productArray, p => ((Product)p).ProductId == searchId);

            stopwatch.Stop();

            if (linearResult != -1)

            {

                Console.WriteLine($"Product found at index {linearResult}: {productArray[linearResult]}");

            }

            else

            {

                Console.WriteLine("Product not found.");

            }

            Console.WriteLine($"Linear search took {stopwatch.ElapsedTicks} ticks");

            // Demo: Binary Search

            Console.WriteLine("\n=== Binary Search Demo ===");

            stopwatch.Restart();

            int binaryResult = SearchAlgorithms.BinarySearch(

                sortedByIdArray,

                p => ((Product)p).ProductId == searchId,

                (p1, p2) => ((Product)p1).ProductId.CompareTo(((Product)p2).ProductId)

            );

            stopwatch.Stop();

            if (binaryResult != -1)

            {

                Console.WriteLine($"Product found at index {binaryResult}: {sortedByIdArray[binaryResult]}");

            }

            else

            {

                Console.WriteLine("Product not found.");

            }

            Console.WriteLine($"Binary search took {stopwatch.ElapsedTicks} ticks");

            // Analysis

            Console.WriteLine("\n=== Time Complexity Analysis ===");

            Console.WriteLine("Linear Search: O(n) - Must check each element in the worst case");

            Console.WriteLine("Binary Search: O(log n) - Divides search space in half each time");

            Console.WriteLine("\nBinary search is more efficient for large datasets, but requires sorted data.");

            Console.WriteLine("Linear search works on unsorted data and is simpler to implement.");

            Console.WriteLine("For small datasets, the difference in performance is negligible.");

            Console.WriteLine("For our e-commerce platform with potentially millions of products,");

            Console.WriteLine("binary search would be more suitable for ID-based searches on indexed fields.");

            Console.ReadKey();

        }

    }

}