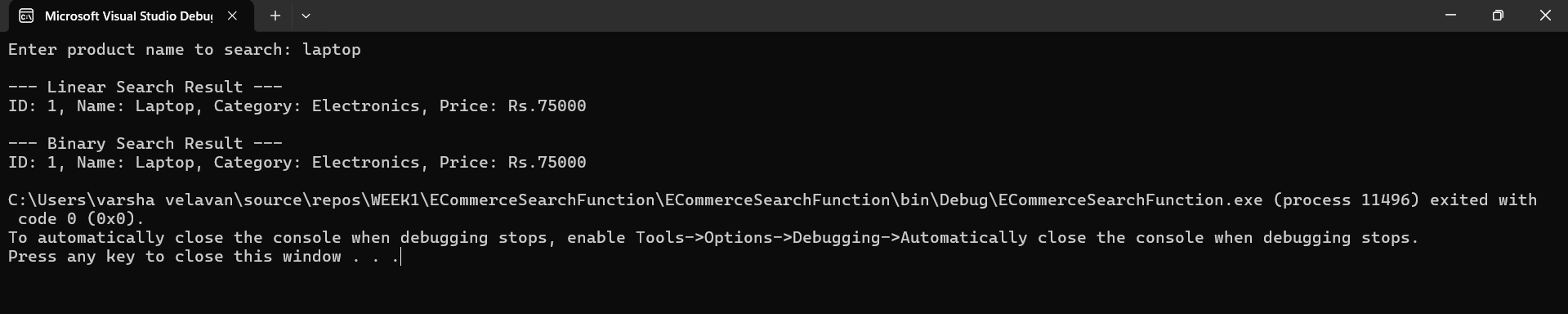
**Week 1: Algorithms\_ Data Structures**

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

using System;  
  
namespace ECommerceSearchFunction  
{  
 public class Product  
 {  
 public int ProductId;  
 public string ProductName;  
 public string Category;  
 public double Price;  
  
 public Product(int id, string name, string category, double price)  
 {  
 ProductId = id;  
 ProductName = name;  
 Category = category;  
 Price = price;  
 }  
 }  
  
 class Program  
 {  
  
 public static Product LinearSearch(Product[] products, string searchName)  
 {  
 foreach (Product product in products)  
 {  
 if (product.ProductName.Equals(searchName, StringComparison.OrdinalIgnoreCase))  
 return product;  
 }  
 return null;  
 }  
  
  
 public static Product BinarySearch(Product[] products, string searchName)  
 {  
 int left = 0;  
 int right = products.Length - 1;  
  
 while (left <= right)  
 {  
 int mid = (left + right) / 2;  
 int compare = string.Compare(products[mid].ProductName, searchName, StringComparison.OrdinalIgnoreCase);  
  
 if (compare == 0)  
 return products[mid];  
 else if (compare < 0)  
 left = mid + 1;  
 else  
 right = mid - 1;  
 }  
  
 return null;  
 }  
  
  
 public static void PrintProduct(Product p)  
 {  
 if (p != null)  
 Console.WriteLine($"ID: {p.ProductId}, Name: {p.ProductName}, Category: {p.Category}, Price: Rs.{p.Price}");  
 else  
 Console.WriteLine("Product not found.");  
 }  
  
 static void Main(string[] args)  
 {  
 Product[] inventory = {  
 new Product(1, "Laptop", "Electronics", 75000),  
 new Product(2, "Mobile", "Electronics", 25000),  
 new Product(3, "Book", "Education", 500),  
 new Product(4, "Desk", "Furniture", 3000)  
 };  
  
   
 Array.Sort(inventory, (x, y) => x.ProductName.CompareTo(y.ProductName));  
  
 Console.Write("Enter product name to search: ");  
 string searchTerm = Console.ReadLine();

Console.WriteLine("\n--- Linear Search Result ---");  
 Product result1 = LinearSearch(inventory, searchTerm);  
 PrintProduct(result1);  
  
 Console.WriteLine("\n--- Binary Search Result ---");  
 Product result2 = BinarySearch(inventory, searchTerm);  
 PrintProduct(result2);  
 }  
 }  
}

**Output:**



**Exercise 7: Financial Forecasting**

**Code:**

using System;  
  
namespace FinancialForecasting  
{  
 class Program  
 {  
   
 static double PredictFutureValue(double currentValue, double growthRate, int years)  
 {  
 if (years == 0)  
 return currentValue;  
 return PredictFutureValue(currentValue, growthRate, years - 1) \* (1 + growthRate);  
 }  
  
 static void Main(string[] args)  
 {  
 Console.WriteLine("Enter current value :");  
 double currentValue = Convert.ToDouble(Console.ReadLine());  
  
 Console.WriteLine("Enter annual growth rate (in %):");  
 double ratePercent = Convert.ToDouble(Console.ReadLine());  
  
 Console.WriteLine("Enter number of years to forecast:");  
 int years = Convert.ToInt32(Console.ReadLine());  
  
 double growthRate = ratePercent / 100;  
  
 double futureValue = PredictFutureValue(currentValue, growthRate, years);  
 Console.WriteLine($"\nPredicted value after {years} years: Rs{futureValue:F2}");  
 }  
 }  
}

**Output:**

