**Cognizant Deep Skilling Week-1**

1.Data Structures and Algorithms

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

CODE:

namespace EcommerceSearch

{

    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;

        }

        public override string ToString()

        {

            return $"{ProductId} - {ProductName} ({Category})";

        }

    }

}

using System;

using System.Linq;

namespace EcommerceSearch

{

    class Program

    {

        static void Main(string[] args)

        {

            Product[] products = new Product[]

            {

                new Product(101, "Laptop", "Electronics"),

                new Product(202, "Shoes", "Fashion"),

                new Product(303, "Mobile", "Electronics"),

                new Product(404, "Watch", "Accessories"),

                new Product(505, "T-Shirt", "Fashion")

            };

            Console.WriteLine("Linear Search:");

            Product result1 = LinearSearch(products, "Mobile");

            Console.WriteLine(result1 != null ? result1.ToString() : "Product not found.");

            Product[] sortedProducts = products.OrderBy(p => p.ProductName).ToArray();

            Console.WriteLine("\nBinary Search:");

            Product result2 = BinarySearch(sortedProducts, "Mobile");

            Console.WriteLine(result2 != null ? result2.ToString() : "Product not found.");

        }

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

        {

            foreach (var product in products)

            {

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

                    return product;

            }

            return null;

        }

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

        {

            int low = 0;

            int high = products.Length - 1;

            while (low <= high)

            {

                int mid = (low + high) / 2;

                int compare = string.Compare(products[mid].ProductName, name, StringComparison.OrdinalIgnoreCase);

                if (compare == 0)

                    return products[mid];

                else if (compare < 0)

                    low = mid + 1;

                else

                    high = mid - 1;

            }

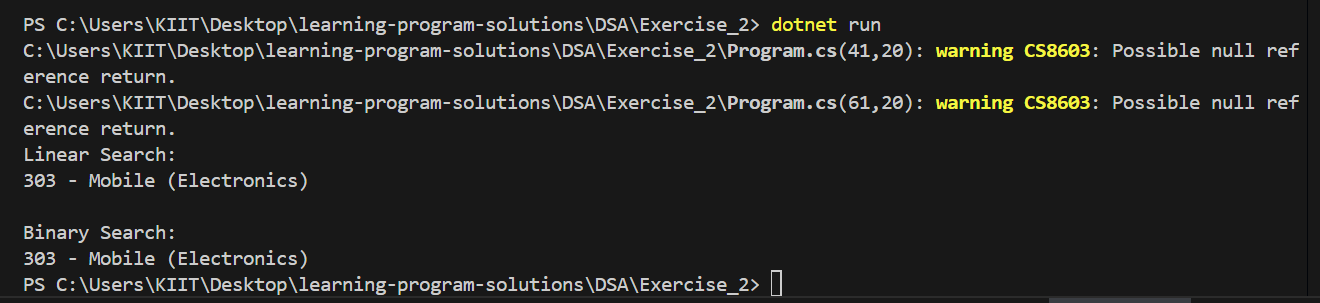
            return null;

        }

    }

}

OUTPUT:



Exercise 7:**Financial Forecasting**

CODE:

using System;

namespace FinancialForecasting

{

    class Program

    {

        static void Main(string[] args)

        {

            double presentValue = 10000;

            double growthRate = 0.05;

            int years = 10;

            double futureValue = Forecast(presentValue, growthRate, years);

            Console.WriteLine($"Forecasted value after {years} years: {futureValue:F2}");

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

            double futureMemo = ForecastMemo(presentValue, growthRate, years, memo);

            Console.WriteLine($"(Memoized) Forecasted value after {years} years: {futureMemo:F2}");

        }

        static double Forecast(double presentValue, double rate, int years)

        {

            if (years == 0)

                return presentValue;

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

        }

        static double ForecastMemo(double presentValue, double rate, int years, double[] memo)

        {

            if (years == 0)

                return presentValue;

            if (memo[years] != 0)

                return memo[years];

            memo[years] = (1 + rate) \* ForecastMemo(presentValue, rate, years - 1, memo);

            return memo[years];

        }

    }

}

OUTPUT:

