EXERCISE – 2 – DSA

E-commerce Platform Search Function

```
using System;

public class Product
{
   public int Id { get; }
   public string Name { get; }
   public string Category { get; }

   public Product(int id, string name, string category)
   {
      Id = id;
      Name = name;
      Category = category;
   }
}

public static class ShopSearch
{
   public static Product LookForProductLinear(Product[] all_products, int target_id)
   {
      foreach (var product_item in all_products)
      {
         if (product_item.Id == target_id)
            return product_item;
      }
```

```
        return null;
    }

    public static Product LookForProductBinary(Product[] sorted_products, int target_id)
    {
        int left_index = 0;
        int right_index = sorted_products.Length - 1;

        while (left_index <= right_index)
        {
            int middle_index = left_index + (right_index - left_index) / 2;
            var current_product = sorted_products[middle_index];

            if (current_product.Id == target_id)
                return current_product;

            if (current_product.Id < target_id)
                left_index = middle_index + 1;
            else
                right_index = middle_index - 1;
        }
        return null;
    }
}

class Program
{
    static void Main()
```

```
{
    var product_list = new[]
    {
        new Product(102, "Wireless Mouse", "Electronics"),
        new Product(205, "Running Shoes", "Sports"),
        new Product(87, "Coffee Maker", "Home"),
        new Product(301, "Novel", "Books")
    };

    var sorted_products = new[]
    {
        new Product(87, "Coffee Maker", "Home"),
        new Product(102, "Wireless Mouse", "Electronics"),
        new Product(205, "Running Shoes", "Sports"),
        new Product(301, "Novel", "Books")
    };

    var search_target = 205;
    var found_item = ShopSearch.LookForProductLinear(product_list, search_target);
    Console.WriteLine(found_item != null
        ? $"[Linear Search] Found: {found_item.Name}"
        : "[Linear Search] Item not found");

    search_target = 102;
    found_item = ShopSearch.LookForProductBinary(sorted_products, search_target);
    Console.WriteLine(found_item != null
        ? $"[Binary Search] Found: {found_item.Name}"
        : "[Binary Search] Item not found");
```

```
    }
}
```

OUTPUT:

```
[Linear Search] Found: Running Shoes
[Binary Search] Found: Wireless Mouse
```

EXERCISE – 7 – DSA

Financial Forecasting

```
using System;


class Financial_Forecast
{
    public static double Calculate_Future_Value_Recursive(double start_amount, double growth_percent, int time_periods)
    {
        if (time_periods == 0)
        {
            return start_amount;
        }
        else
        {
            return Calculate_Future_Value_Recursive(start_amount, growth_percent, time_periods - 1) * (1 + growth_percent);
        }
    }


    public static double Calculate_Future_Value_Iterative(double start_amount, double growth_percent, int time_periods)
    {
```

```
    double current_value = start_amount;
    for (int count = 0; count < time_periods; count++)
    {
      current_value *= (1 + growth_percent);
    }
    return current_value;
  }

  static void Main()
  {
    double initial_investment = 1000;
    double annual_growth = 0.05;
    int investment_years = 3;

    double result_recursive = Calculate_Future_Value_Recursive(initial_investment, annual_growth, investment_years);
    Console.WriteLine($"[Recursive Method] Future value: ${result_recursive:F2}");

    double result_iterative = Calculate_Future_Value_Iterative(initial_investment, annual_growth, investment_years);
    Console.WriteLine($"[Iterative Method] Future value: ${result_iterative:F2}");
  }
}
```

OUTPUT

```
[Recursive Method] Future value: $1157.63
[Iterative Method] Future value: $1157.63
```