**WEEK 1  
  
Data structures and Algorithms  
  
Exercise 2: E-commerce Platform Search Function**

**CODE  
  
-Product.cs  
  
 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;**

**}**

**}**

**-Program.cs**

**using System;**

**using System.Linq;**

**class Program**

**{**

**static void Main()**

**{**

**Product[] products = {**

**new Product(1, "Shoes", "Footwear"),**

**new Product(2, "Shirt", "Clothing"),**

**new Product(3, "Watch", "Accessories"),**

**new Product(4, "Jeans", "Clothing"),**

**new Product(5, "Laptop", "Electronics")**

**};**

**Console.WriteLine("🔍 Linear Search:");**

**var result1 = LinearSearch(products, "Watch");**

**Console.WriteLine(result1 != null ? $"Found: {result1.ProductName}" : "Not found");**

**Console.WriteLine("\n🔍 Binary Search:");**

**var sortedProducts = products.OrderBy(p => p.ProductName).ToArray();**

**var result2 = BinarySearch(sortedProducts, "Watch");**

**Console.WriteLine(result2 != null ? $"Found: {result2.ProductName}" : "Not found");**

**Console.WriteLine("\n📈 Financial Forecast:");**

**double current = 1000;**

**double growth = 0.10;**

**int years = 5;**

**double future = ForecastFutureValue(current, growth, years);**

**Console.WriteLine($"Future Value after {years} years: ₹{future:F2}");**

**}**

**static Product? LinearSearch(Product[] list, string name)**

**{**

**foreach (var product in list)**

**{**

**if (product.ProductName == name)**

**return product;**

**}**

**return null;**

**}**

**static Product? BinarySearch(Product[] list, string name)**

**{**

**int low = 0, high = list.Length - 1;**

**while (low <= high)**

**{**

**int mid = (low + high) / 2;**

**int compare = string.Compare(name, list[mid].ProductName);**

**if (compare == 0)**

**return list[mid];**

**else if (compare < 0)**

**high = mid - 1;**

**else**

**low = mid + 1;**

**}**

**return null;**

**}**

**static double ForecastFutureValue(double currentValue, double growthRate, int years)**

**{**

**if (years == 0)**

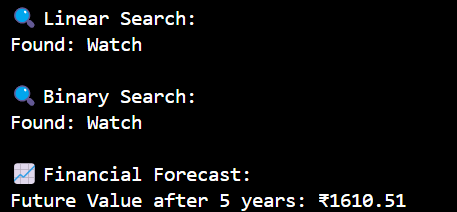
**return currentValue;**

**return ForecastFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);**

**}**

**}**

**OUTPUT**

****

**Exercise 7: Financial Forecasting  
  
CODE-  
  
-Forecast.cs  
  
public class Forecast**

**{**

**public double CalculateFutureValueRecursive(double currentValue, double growthRate, int years)**

**{**

**if (years == 0)**

**return currentValue;**

**return CalculateFutureValueRecursive(currentValue \* (1 + growthRate), growthRate, years - 1);**

**}**

**public double CalculateFutureValueIterative(double currentValue, double growthRate, int years)**

**{**

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

**{**

**currentValue \*= (1 + growthRate);**

**}**

**return currentValue;**

**}**

**}**

**-Program.cs**

**using System;**

**class Program**

**{**

**static void Main()**

**{**

**Forecast forecast = new Forecast();**

**double presentValue = 1000;**

**double annualGrowthRate = 0.10;**

**int numberOfYears = 5;**

**Console.WriteLine("📊 Financial Forecast Using Recursion:");**

**double futureRecursive = forecast.CalculateFutureValueRecursive(presentValue, annualGrowthRate, numberOfYears);**

**Console.WriteLine($"Recursive Future Value after {numberOfYears} years: ₹{futureRecursive:F2}");**

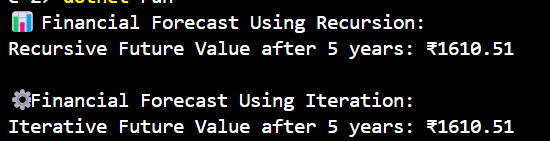
**Console.WriteLine("\n⚙️ Financial Forecast Using Iteration:");**

**double futureIterative = forecast.CalculateFutureValueIterative(presentValue, annualGrowthRate, numberOfYears);**

**Console.WriteLine($"Iterative Future Value after {numberOfYears} years: ₹{futureIterative:F2}");**

**}**

**}**

**OUTPUT-  
  
**