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

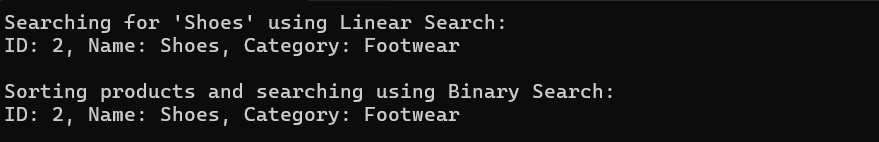
**Scenario:**

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

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.\
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Output:**

****

**Code:**

**Program.cs-**

using System;

using System.Collections.Generic;

namespace ECommerceSearch

{

class Program

{

static void Main(string[] args)

{

List<Product> products = new List<Product>

{

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

new Product(2, "Shoes", "Footwear"),

new Product(3, "Smartphone", "Electronics"),

new Product(4, "Notebook", "Stationery"),

new Product(5, "Pen", "Stationery")

};

Console.WriteLine("Searching for 'Shoes' using Linear Search:");

var linearResult = LinearSearch(products, "Shoes");

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

Console.WriteLine("\nSorting products and searching using Binary Search:");

products.Sort((x, y) => x.ProductName.CompareTo(y.ProductName));

var binaryResult = BinarySearch(products, "Shoes");

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

}

static Product LinearSearch(List<Product> products, string name)

{

foreach (var product in products)

{

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

return product;

}

return null;

}

static Product BinarySearch(List<Product> products, string name)

{

int left = 0;

int right = products.Count - 1;

while (left <= right)

{

int mid = (left + right) / 2;

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

if (comparison == 0)

return products[mid];

else if (comparison < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

}

}

**Product.cs-**

namespace ECommerceSearch

{

public class Product

{

public int ProductId { get; set; }

public string ProductName { get; set; }

public string Category { get; set; }

public Product(int productId, string productName, string category)

{

ProductId = productId;

ProductName = productName;

Category = category;

}

public override string ToString()

{

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

}

}

}