**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.

# THE CODE:

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})";

}

}

public class SearchService

{

// Linear Search: O(n)

public static Product LinearSearch(Product[] products, int productId)

{

foreach (var product in products)

{

if (product.ProductId == productId)

return product;

}

return null;

}

// Binary Search: O(log n) – requires sorted array

public static Product BinarySearch(Product[] sortedProducts, int productId)

{

int low = 0;

int high = sortedProducts.Length - 1;

while (low <= high)

{

int mid = (low + high) / 2;

if (sortedProducts[mid].ProductId == productId)

return sortedProducts[mid];

else if (sortedProducts[mid].ProductId < productId)

low = mid + 1;

else

high = mid - 1;

}

return null;

}

}

class Program

{

static void Main(string[] args)

{

// Create sample product data

Product[] products = new Product[]

{

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

new Product(105, "Shoes", "Fashion"),

new Product(110, "Book", "Stationery"),

new Product(120, "Phone", "Electronics"),

new Product(130, "Desk", "Furniture")

};

Console.WriteLine(" Linear Search:");

var result1 = SearchService.LinearSearch(products, 120);

Console.WriteLine(result1 != null ? $"Found: {result1}" : "Product not found");

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

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

var result2 = SearchService.BinarySearch(sortedProducts, 120);

Console.WriteLine(result2 != null ? $"Found: {result2}" : "Product not found");

Console.WriteLine("\n Time Complexity:");

Console.WriteLine("Linear Search: O(n)");

Console.WriteLine("Binary Search: O(log n) — but requires sorted data");

}

}

# THE OUTPUT:

