**Week - 1**

**Name -** Rakshit Ranjan

**SuperSet\_ID-** 6363335

**Q1 : E-commerce Platform Search Function**

**Code :-**

using System;

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;

}

}

class Program

{

public static Product LinearSearch(Product[] products, string searchTerm)

{

foreach (var product in products)

{

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

return product;

}

return null;

}

public static Product BinarySearch(Product[] products, string searchTerm)

{

int left = 0, right = products.Length - 1;

while (left <= right)

{

int mid = (left + right) / 2;

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

if (compareResult == 0)

return products[mid];

else if (compareResult < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

static void Main()

{

Product[] products = new Product[]

{

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

new Product(2, "Phone", "Electronics"),

new Product(3, "Shoes", "Footwear"),

new Product(4, "Tablet", "Electronics"),

new Product(5, "Watch", "Accessories"),

new Product(6, "Audio", "Electronics")

};

Array.Sort(products, (p1, p2) => p1.ProductName.CompareTo(p2.ProductName));

Console.WriteLine("E-Commerce Product Search");

Console.WriteLine("Choose search method:");

Console.WriteLine("1. Linear Search (O(n))");

Console.WriteLine("2. Binary Search (O(log n)) - Search by exact product name only");

Console.Write("Enter choice (1 or 2): ");

int choice = int.Parse(Console.ReadLine());

Console.Write("\nEnter search keyword: ");

string keyword = Console.ReadLine();

Product result = null;

if (choice == 1)

result = LinearSearch(products, keyword);

else if (choice == 2)

result = BinarySearch(products, keyword);

else

{

Console.WriteLine("Invalid choice.");

return;

}

if (result != null)

Console.WriteLine($"\nProduct Found:\nID: {result.ProductId}, Name: {result.ProductName}, Category: {result.Category}");

else

Console.WriteLine("\nNo product found with exact name.");

/\*

\* Time Complexity Analysis:

\* Linear Search: O(n) — Best: O(1), Avg: O(n/2), Worst: O(n)

\* Binary Search: O(log n) — Best: O(1), Avg: O(log n), Worst: O(log n)

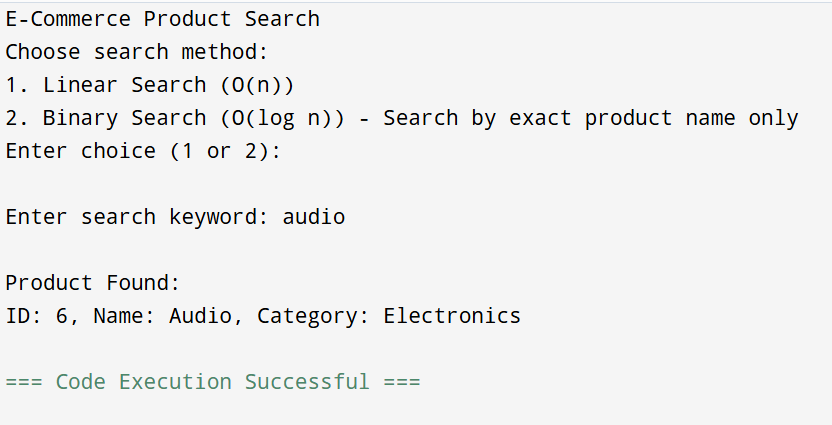
\* Binary search is faster but only works on sorted data.

\*/

}

}

**Output :-**

****

**Q2 : Financial Forecasting**

**Code :-**

using System;

class FinancialForecast

{

static void Main()

{

Console.Write("Enter initial investment amount: ");

if (!double.TryParse(Console.ReadLine(), out double initialValue) || initialValue < 0)

{

Console.WriteLine("Invalid input. Please enter a valid positive number.");

return;

}

Console.Write("Enter annual growth rate (in %): ");

if (!double.TryParse(Console.ReadLine(), out double growthRatePercent))

{

Console.WriteLine("Invalid input. Please enter a valid number.");

return;

}

double growthRate = growthRatePercent / 100.0;

Console.Write("Enter number of years to forecast: ");

if (!int.TryParse(Console.ReadLine(), out int years) || years < 0)

{

Console.WriteLine("Invalid input. Please enter a valid positive integer.");

return;

}

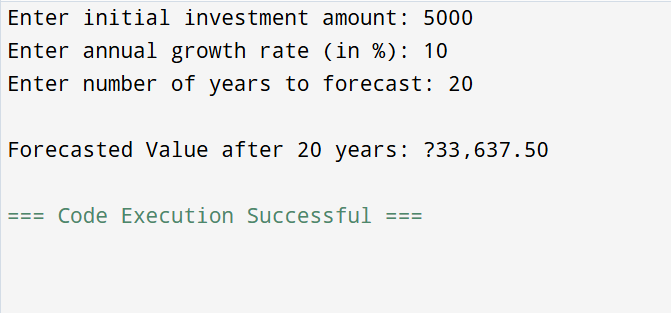
double futureValue = initialValue \* Math.Pow(1 + growthRate, years);

Console.WriteLine($"\nForecasted Value after {years} years: {futureValue:C2}");

}

}

**Output :-**

****