Algorithms Data Structures

**Exercise 1: Inventory Management System**

**CODE**

using System;

class Productnew

{

    private int productId;

    private string productName;

    private string category;

    private static Productnew[] products = new Productnew[100];

    private static int index = 0;

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

    {

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public static void AddProduct(Productnew product)

    {

        products[index++] = product;

    }

    public int GetProductId()

    {

        return productId;

    }

    public static Productnew LinearSearch(int productId)

    {

        for (int i = 0; i < index; i++)

        {

            if (products[i].productId == productId)

                return products[i];

        }

        return null;

    }

    public static Productnew BinarySearch(int productId)

    {

        int low = 0;

        int high = index - 1;

        while (low <= high)

        {

            int mid = (low + high) / 2;

            if (products[mid].productId == productId)

                return products[mid];

            else if (productId > products[mid].productId)

                low = mid + 1;

            else

                high = mid - 1;

        }

        return null;

    }

    public static Productnew[] GetAllProducts()

    {

        return products;

    }

    public static int GetIndex()

    {

        return index;

    }

    public override string ToString()

    {

        return $"Id: {productId}    Name: {productName}    Category: {category}";

    }

}

class Program

{

    static void Main(string[] args)

    {

        Productnew p4 = new Productnew(4, "Tea", "Dairy Product");

        Productnew p2 = new Productnew(2, "Paneer", "Dairy Product");

        Productnew p1 = new Productnew(1, "Milk", "Dairy Product");

        Productnew p3 = new Productnew(3, "Coffee", "Dairy Product");

        Console.WriteLine(p1);

        Console.WriteLine(p2);

        Console.WriteLine(p3);

        Console.WriteLine(p4);

        Productnew.AddProduct(p4);

        Productnew.AddProduct(p2);

        Productnew.AddProduct(p3);

        Productnew.AddProduct(p1);

        Array.Sort(Productnew.GetAllProducts(), 0, Productnew.GetIndex(),

            Comparer<Productnew>.Create((a1, a2) => a1.GetProductId().CompareTo(a2.GetProductId()))

        );

        Console.WriteLine("\n-----AFTER SORTING-------\n");

        for (int i = 0; i < Productnew.GetIndex(); i++)

        {

            Console.WriteLine(Productnew.GetAllProducts()[i]);

        }

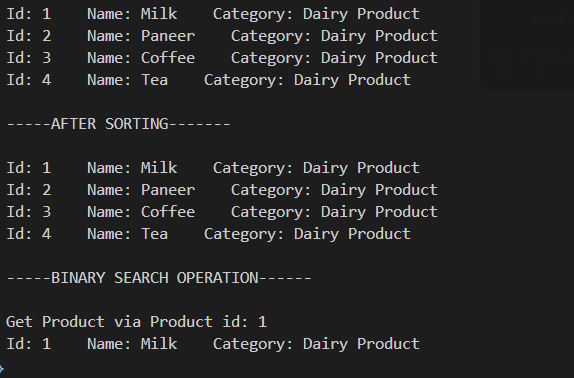
        Console.WriteLine("\n-----BINARY SEARCH OPERATION------\n");

        Console.WriteLine("Get Product via Product id: 1\n" + Productnew.BinarySearch(1));

    }

}

**OUTPUT**

****

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

**CODE**

using System;

using System.Collections.Generic;

class Product

{

    private int productId, quantity;

    private string productName;

    private double price;

    private static Dictionary<int, Product> store = new Dictionary<int, Product>();

    public Product(int productId, string productName, int quantity, double price)

    {

        this.productId = productId;

        this.productName = productName;

        this.quantity = quantity;

        this.price = price;

    }

    public static string AddProduct(Product newProduct)

    {

        int oldStoreSize = store.Count;

        store[newProduct.productId] = newProduct;

        int newStoreSize = store.Count;

        return newStoreSize > oldStoreSize ? "Product Added Successfully" : "Failed Add operation";

    }

    public static Product RemoveProduct(int productId)

    {

        if (store.ContainsKey(productId))

        {

            Product removed = store[productId];

            store.Remove(productId);

            return removed;

        }

        return null;

    }

    public static Product UpdateProduct(int productId, string productName, double price, int quantity)

    {

        if (!store.ContainsKey(productId))

            return null;

        Product updatedProduct = store[productId];

        if (price >= 0) updatedProduct.price = price;

        if (quantity >= 0) updatedProduct.quantity = quantity;

        if (!string.IsNullOrEmpty(productName) && productName != updatedProduct.productName)

        {

            updatedProduct.productName = productName;

        }

        return updatedProduct;

    }

    public string GetProductName() => productName;

    public double GetProductPrice() => price;

    public int GetProductQuantity() => quantity;

    public override string ToString()

    {

        return $"ID: {productId}, Name: {productName}, Price: {price}, Qty: {quantity}";

    }

}

class IMS

{

    static void Main(string[] args)

    {

        Product product = new Product(1, "rice", 10000, 20.23);

        Console.WriteLine(product);

        string status = Product.AddProduct(product);

        Console.WriteLine(status);

        Product updated = Product.UpdateProduct(1, "basmati rice", -1, -1);

        if (updated != null)

        {

            Console.WriteLine(updated);

        }

        else

        {

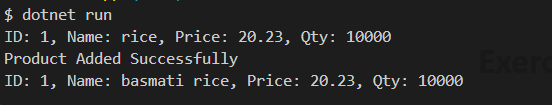
            Console.WriteLine("Product not found for update.");

        }

    }

}

**OUTPUT**

****

**Exercise 3: Sorting Customer Orders**

**CODE**

using System;

class Order

{

    private int orderId;

    private string customerName;

    private double totalPrice;

    private static Order[] orders = new Order[10];

    private static int index = 0;

    public Order(int orderId, string customerName, double totalPrice)

    {

        this.orderId = orderId;

        this.customerName = customerName;

        this.totalPrice = totalPrice;

    }

    public static void AddOrder(Order order)

    {

        orders[index++] = order;

    }

    public static void BubbleSort()

    {

        for (int i = 0; i < index; i++)

        {

            for (int j = 0; j < index - i - 1; j++)

            {

                if (orders[j].totalPrice > orders[j + 1].totalPrice)

                {

                    Swap(j, j + 1);

                }

            }

        }

    }

    public static void QuickSort(int low, int high)

    {

        if (low < high)

        {

            int pi = Partition(low, high);

            QuickSort(low, pi - 1);

            QuickSort(pi + 1, high);

        }

    }

    private static int Partition(int low, int high)

    {

        Order pivot = orders[high];

        int i = low - 1;

        for (int j = low; j <= high - 1; j++)

        {

            if (orders[j].totalPrice < pivot.totalPrice)

            {

                i++;

                Swap(i, j);

            }

        }

        Swap(i + 1, high);

        return i + 1;

    }

    private static void Swap(int i, int j)

    {

        Order temp = orders[i];

        orders[i] = orders[j];

        orders[j] = temp;

    }

    public static Order[] GetAllOrders() => orders;

    public static int GetIndex() => index;

    public override string ToString()

    {

        return $"ID: {orderId}    Customer Name: {customerName}    Total Price: {totalPrice}";

    }

}

class SCO

{

    static void Main()

    {

        Order o1 = new Order(1, "Sujay", 1220.98);

        Order o2 = new Order(2, "Shreya", 1200.98);

        Order o3 = new Order(3, "Rishav", 3220.98);

        Order o4 = new Order(4, "John", 1000.98);

        Order o5 = new Order(5, "Noah", 1120.98);

        Order.AddOrder(o1);

        Order.AddOrder(o2);

        Order.AddOrder(o3);

        Order.AddOrder(o4);

        Order.AddOrder(o5);

        Console.WriteLine("-----All Orders Before Sorting-----\n");

        for (int i = 0; i < Order.GetIndex(); i++)

        {

            Console.WriteLine(Order.GetAllOrders()[i]);

        }

        Console.WriteLine("\n-----After Sorting by Total Price (QuickSort)-----\n");

        Order.QuickSort(0, Order.GetIndex() - 1);

        for (int i = 0; i < Order.GetIndex(); i++)

        {

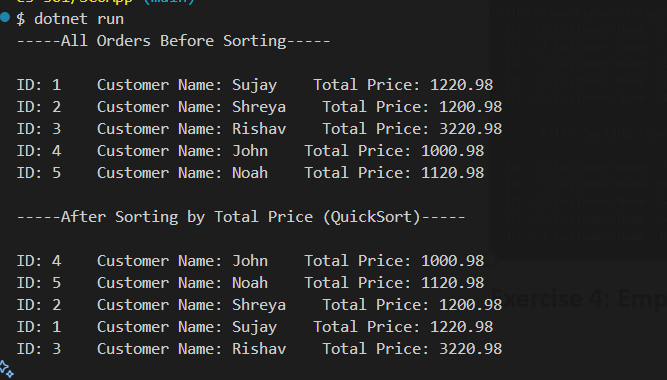
            Console.WriteLine(Order.GetAllOrders()[i]);

        }

    }

}

**OUTPUT**

****

**Exercise 4: Employee Management System**

**CODE**

using System;

using System.Collections.Generic;

class Employee

{

    private int employeeId;

    private string name;

    private string position;

    private double salary;

    private static Employee[] employees = new Employee[100];

    private static int index = 0;

    private static HashSet<int> employeeIds = new HashSet<int>();

    public Employee(int employeeId, string name, string position, double salary)

    {

        if (employeeIds.Contains(employeeId))

            throw new ArgumentException($"Employee Id: {employeeId} already exists!");

        this.employeeId = employeeId;

        this.name = name;

        this.position = position;

        this.salary = salary;

        employeeIds.Add(employeeId);

    }

    public static bool Add(Employee employee)

    {

        employees[index++] = employee;

        return true;

    }

    public static Employee Search(int employeeId)

    {

        for (int i = 0; i < index; i++)

        {

            if (employees[i] != null && employees[i].employeeId == employeeId)

                return employees[i];

        }

        return null;

    }

    public static void Traverse()

    {

        for (int i = 0; i < index; i++)

        {

            if (employees[i] != null)

                Console.WriteLine(employees[i]);

        }

    }

    public static string Delete(int employeeId)

    {

        for (int i = 0; i < index; i++)

        {

            if (employees[i] != null && employees[i].employeeId == employeeId)

            {

                Employee deleted = employees[i];

                employees[i] = null;

                employeeIds.Remove(employeeId);

                return $"{deleted}   Details Deleted Successfully";

            }

        }

        return "Invalid Employee Id";

    }

    public string GetEmployeeName() => name;

    public string GetPosition() => position;

    public double GetSalary() => salary;

    public int GetEmployeeId() => employeeId;

    public override string ToString()

    {

        return $"ID: {employeeId}    Name: {name}    Position: {position}    Salary: {salary}";

    }

}

class EMS

{

    static void Main()

    {

        Employee employee = new Employee(1, "Sujay Kumar", "Full-Stack Developer", 400000);

        Employee employee1 = new Employee(2, "Rishav Kumar", "Mobile App Developer", 200000);

        Employee employee2 = new Employee(3, "Abhishek Roy", "ML Engineer", 800000);

        Console.WriteLine(employee);

        Console.WriteLine(employee1);

        Console.WriteLine(employee2);

        Console.WriteLine("\n------Adding all Employees to the employees List--------\n");

        Console.WriteLine(Employee.Add(employee));

        Console.WriteLine(Employee.Add(employee1));

        Console.WriteLine(Employee.Add(employee2));

        Console.WriteLine("\n-----All Employees added Successfully-----\n");

        Console.WriteLine("Search Employee whose Id is : 2");

        var searched = Employee.Search(2);

        Console.WriteLine(searched != null ? searched.ToString() : "Employee not found");

        Console.WriteLine("\n------Traverse the employees List------\n");

        Employee.Traverse();

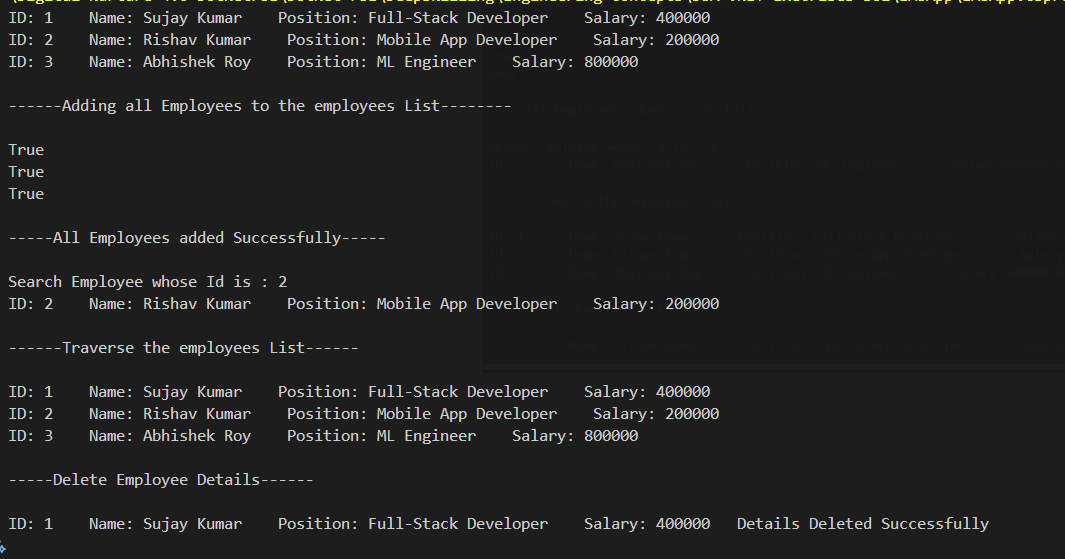
        Console.WriteLine("\n-----Delete Employee Details------\n");

        Console.WriteLine(Employee.Delete(1));

    }

}

**OUTPUT**

****

**Exercise 6: Library Management System**

**Code**

using System;

using System.Collections.Generic;

class Book

{

    private int bookId;

    private string title;

    private string author;

    private static List<Book> books = new List<Book>();

    public Book(int bookId, string title, string author)

    {

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    public static void Add(Book book)

    {

        books.Add(book);

    }

    public static Book Search(int bookId)

    {

        foreach (var book in books)

        {

            if (book.bookId == bookId)

                return book;

        }

        return null;

    }

    public static Book BinarySearch(int bookId)

    {

        books.Sort((b1, b2) => b1.bookId.CompareTo(b2.bookId)); // Ensure list is sorted

        int low = 0;

        int high = books.Count - 1;

        while (low <= high)

        {

            int mid = (low + high) / 2;

            if (books[mid].bookId == bookId)

                return books[mid];

            else if (bookId < books[mid].bookId)

                high = mid - 1;

            else

                low = mid + 1;

        }

        return null;

    }

    public override string ToString()

    {

        return $"Id: {bookId}    Title: {title}    Author: {author}";

    }

}

class LMS

{

    static void Main()

    {

        Book book1 = new Book(1, "Ikigai", "Robert Frost");

        Book book2 = new Book(2, "Rich Dad Poor Dad", "Frost");

        Book.Add(book1);

        Book.Add(book2);

        Console.WriteLine("All Books:");

        foreach (var book in Book.BinarySearch(2) != null ? Book.BinarySearch(2).ToString() : "Book not found")

        {

            Console.WriteLine(book1);

            Console.WriteLine(book2);

        }

        Console.WriteLine("\nSearch by Book ID 1:");

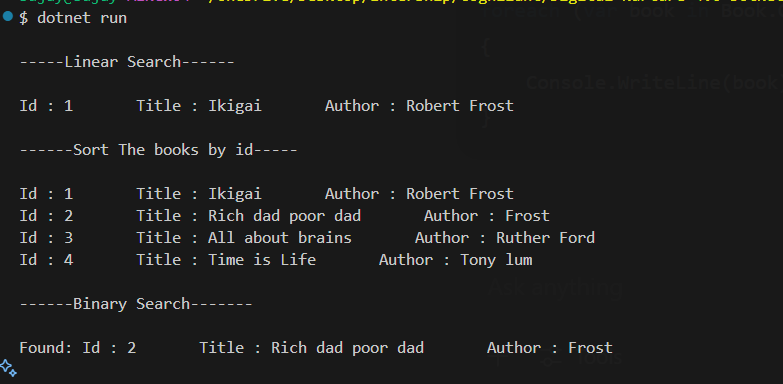
        var searchResult = Book.Search(1);

        Console.WriteLine(searchResult != null ? searchResult.ToString() : "Book not found");

    }

}

**OUTPUT**

****

**Exercise 7: Financial Forecasting**

**CODE**

using System;

using System.Collections.Generic;

using System.Linq;

class FinancialForcasting

{

    private int year;

    private double rate;

    private double principle;

    public FinancialForcasting(double principle, double rate, int year)

    {

        this.principle = principle;

        this.rate = rate;

        this.year = year;

    }

    public double FutureValue(int year)

    {

        if (year == 0)

            return principle;

        return FutureValue(year - 1) \* (1 + rate);

    }

    public override string ToString()

    {

        return $"Principle:    {principle}    Year:     {year}     Rate:     {rate}";

    }

}

class Program

{

    static void Main(string[] args)

    {

        FinancialForcasting first = new FinancialForcasting(1000.0, 2.3 / 100, 3);

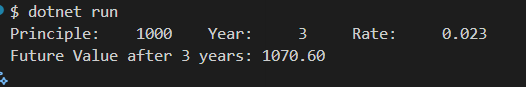
        Console.WriteLine(first);

        Console.WriteLine($"Future Value after 3 years: {first.FutureValue(3):F2}");

    }

}

**OUTPUT**

****