**Data structures and Algorithms**

**Exercise 2: E – Commerce Platform Search Function**

**Code:**

import java.util.\*;

public class EcommercePlatformSearchFunction {

    static class Product {

        int productId;

        String productName;

        String category;

        public Product(int productId, String productName, String category) {

            this.productId = productId;

            this.productName = productName;

            this.category = category;

        }

        public String toString() {

            return productId + " - " + productName + " (" + category + ")";

        }

    }

    public static Product linearSearch(Product[] products, String targetName) {

        for (Product p : products) {

            if (p.productName.equalsIgnoreCase(targetName)) {

                return p;

            }

        }

        return null;

    }

    public static Product binarySearch(Product[] products, String targetName) {

        int low = 0, high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

            int cmp = products[mid].productName.compareToIgnoreCase(targetName);

            if (cmp == 0) return products[mid];

            else if (cmp < 0) low = mid + 1;

            else high = mid - 1;

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shirt", "Fashion"),

            new Product(103, "Book", "Education"),

            new Product(104, "Mouse", "Electronics"),

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

        };

        String target = "Book";

        System.out.println("Searching for: " + target);

        Product foundLinear = linearSearch(products, target);

        System.out.println("Linear Search Result: " + (foundLinear != null ? foundLinear : "Not Found"));

        Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

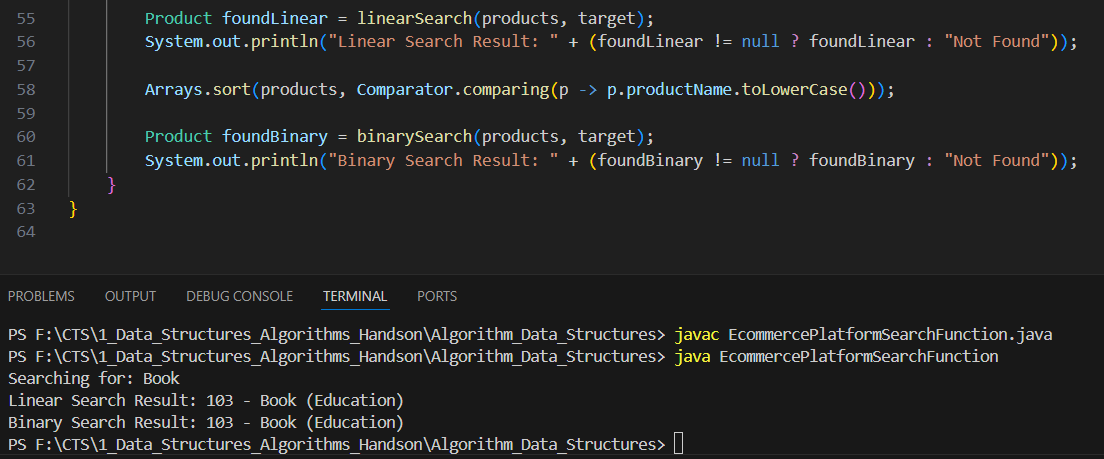
        Product foundBinary = binarySearch(products, target);

        System.out.println("Binary Search Result: " + (foundBinary != null ? foundBinary : "Not Found"));

    }

}

**Output:**

****

**Exercise 7: Financial ForeCasting**

**Code:**

public class FinancialForecasting {

    public static double forecastRecursive(double initial, double rate, int years) {

        if (years == 0) return initial;

        return forecastRecursive(initial, rate, years - 1) \* (1 + rate);

    }

    public static double forecastIterative(double initial, double rate, int years) {

        double result = initial;

        for (int i = 1; i <= years; i++) {

            result \*= (1 + rate);

        }

        return result;

    }

    public static double forecastOptimized(double initial, double rate, int years) {

        return initial \* Math.pow(1 + rate, years);

    }

    public static void main(String[] args) {

        double initialAmount = 30000;

        double annualRate = 0.10;

        int years = 5;

        System.out.printf("Initial Amount: Rs.%.2f\n", initialAmount);

        System.out.println("Growth Rate: 10% per year");

        System.out.println("Time Period: " + years + " years\n");

        double valueRecursive = forecastRecursive(initialAmount, annualRate, years);

        double valueIterative = forecastIterative(initialAmount, annualRate, years);

        double valueOptimized = forecastOptimized(initialAmount, annualRate, years);

        System.out.printf("Recursive Forecast: Rs.%.2f\n", valueRecursive);

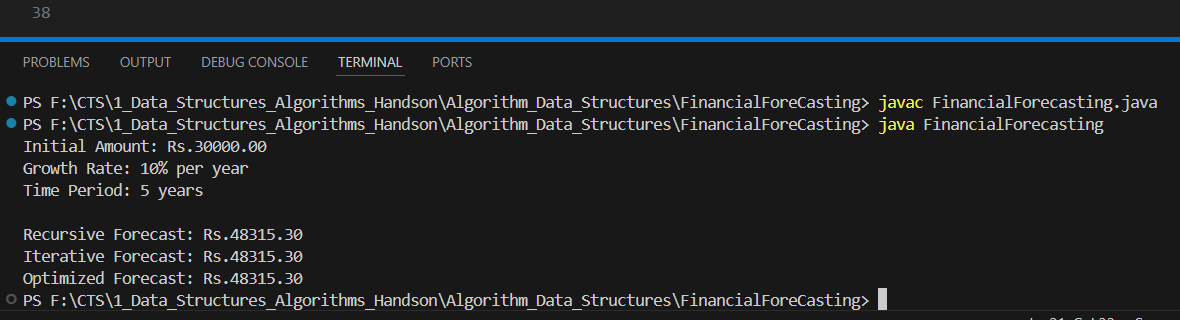
        System.out.printf("Iterative Forecast: Rs.%.2f\n", valueIterative);

        System.out.printf("Optimized Forecast: Rs.%.2f\n", valueOptimized);

    }

}

**Output:**

****