WEEK 1 – ALGORITHMS DATA STRUCTURE

HANDS-ON –1 :- E-COMMERCE PLATFORM SEARCH FUNCTION

**CODE FOR ECOMMERCE PLATFORM SEARCH FUNCTION:- Main.java**

public class Main {

    public static void main(String[] args) {

        Product[] products = {

            new Product(201, "Phone", "Electronics"),

            new Product(202, "Laptop", "Electronics"),

            new Product(203, "Shoes", "Footwear"),

            new Product(204, "Shirt", "Clothing")

        };

        System.out.println(" Linear Search for 'Laptop'");

        int linearIndex = Search.linearSearch(products, "Laptop");

        System.out.println(linearIndex >= 0 ? products[linearIndex] : "Product not found");

        System.out.println("\n Sorting products for Binary Search...");

        Search.sortProductsByName(products);

        System.out.println(" Binary Search for 'Shoes'");

        int binaryIndex = Search.binarySearch(products, "Shoes");

        System.out.println(binaryIndex >= 0 ? products[binaryIndex] : "Product not found");   }

}

**Product.java**

public 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 "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;

    }

}

**Search.java**

import java.util.Arrays;

public class Search {

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

        for (int i = 0; i < products.length; i++) {

            if (products[i].productName.equalsIgnoreCase(targetName)) {

                return i;

            }

        }

        return -1;

    }

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

        int low = 0;

        int high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (compare == 0) return mid;

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

            else high = mid - 1;

        }

        return -1;

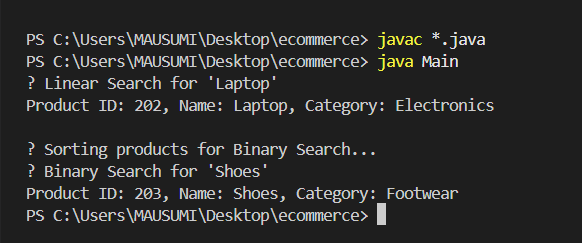
    }

    public static void sortProductsByName(Product[] products) {

        Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

    }

}

**OUTPUT OF ECOMMERCE PLATFORM SEARCH FUNCTION:-**

HANDS-ON – 2 :- FINANCIAL FORECASTING

**CODE FOR FINANCIAL FORECASTING**:-

import java.util.Scanner;

public class FinancialForecast {

    public static double futureValue(double presentValue, double rate, int years) {

        if (years == 0) {

            return presentValue;

        }

        return futureValue(presentValue, rate, years - 1) \* (1 + rate);

    }

    public static double futureValueMemo(double presentValue, double rate, int years, double[] memo) {

        if (years == 0) {

            return presentValue;

        }

        if (memo[years] != 0) {

            return memo[years];

        }

        memo[years] = futureValueMemo(presentValue, rate, years - 1, memo) \* (1 + rate);

        return memo[years];

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter present value: ");

        double presentValue = scanner.nextDouble();

        System.out.print("Enter annual growth rate (e.g., 0.05 for 5%): ");

        double rate = scanner.nextDouble();

        System.out.print("Enter number of years: ");

        int years = scanner.nextInt();

        double result = futureValue(presentValue, rate, years);

        System.out.printf("Future value (recursive): %.2f\n", result);

        double[] memo = new double[years + 1];

        double resultMemo = futureValueMemo(presentValue, rate, years, memo);

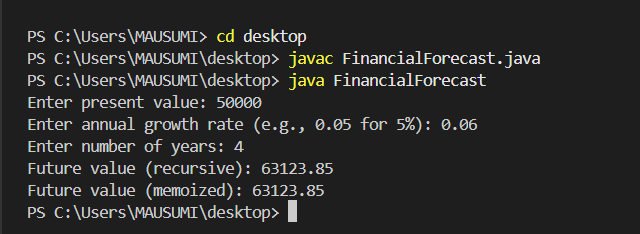
        System.out.printf("Future value (memoized): %.2f\n", resultMemo);

        scanner.close();

    }

}

**OUTPUT OF FINANCIAL FORECASTING:-**

****