**DATA STRUCTURE AND ALGORITHM**

**Exercise 7: Financial Forecasting**

**CODE :**

**FinancialForecast.java**

public class FinancialForecast {

    public static double predictFutureValue(double currentValue, double growthRate, int years) {

        if (years == 0) {

            return currentValue;

        }

        return predictFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);

    }

    public static double predictFutureValueMemo(double currentValue, double growthRate, int years, Double[] memo) {

        if (years == 0) {

            return currentValue;

        }

        if (memo[years] != null) {

            return memo[years];

        }

        memo[years] = predictFutureValueMemo(currentValue \* (1 + growthRate), growthRate, years - 1, memo);

        return memo[years];

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        double initialInvestment = 1000.0;

        double growthRate = 0.1; // 10%

        int years = 5;

        double futureValue = FinancialForecast.predictFutureValue(initialInvestment, growthRate, years);

        System.out.println("Predicted Future Value (Recursive): $" + futureValue);

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

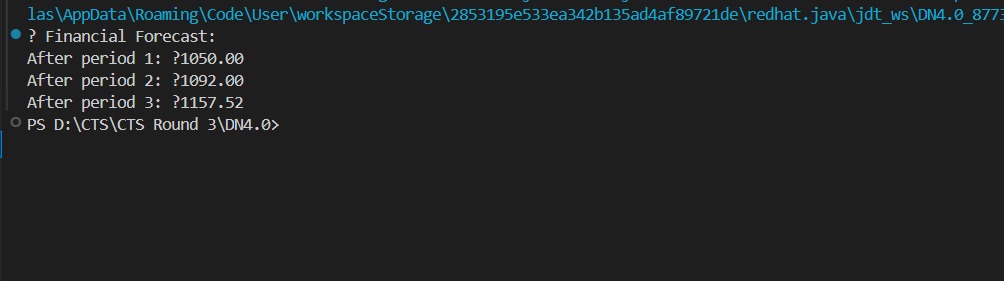
        double optimizedValue = FinancialForecast.predictFutureValueMemo(initialInvestment, growthRate, years, memo);

        System.out.println("Predicted Future Value (Memoized): $" + optimizedValue);

    }

}

**OUTPUT :**

****

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

**CODE :**

**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;

    }

}

**ProductSearch.java**

import java.util.Arrays;

public class ProductSearch {

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

        for (Product product : products) {

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

                return product;

            }

        }

        return null;

    }

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

        int left = 0;

        int right = products.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;

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

            if (comparison == 0) {

                return products[mid];

            } else if (comparison < 0) {

                left = mid + 1;

            } else {

                right = mid - 1;

            }

        }

        return null;

    }

}

**Main.java**

import java.util.Arrays;

import java.util.Comparator;

public class Main {

    public static void main(String[] args) {

        Product[] products = {

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

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

            new Product(103, "Smartphone", "Electronics"),

            new Product(104, "Shoes", "Footwear"),

            new Product(105, "Notebook", "Stationery")

        };

        Product found = ProductSearch.linearSearch(products, "Shoes");

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

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

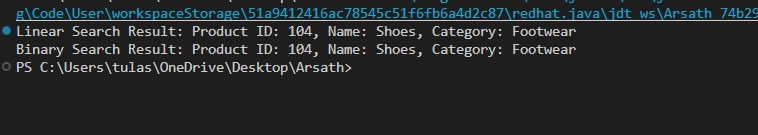
        Product foundBinary = ProductSearch.binarySearch(products, "Shoes");

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

    }

}

**OUTPUT :**

****