**Exercise 7: Financial Forecasting**

**Understand Recursive Algorithm**

**Recursion** is a programming technique where a method calls itself to solve a problem in smaller chunks. It’s especially useful for tasks like financial forecasting, where future values depend on past outcomes.

Program:

public class FinancialForecast {

    public static double forecastValue(double presentValue, double growthRate, int years) {

        if (years == 0) {

            return presentValue;

        }

        return forecastValue(presentValue, growthRate, years - 1) \* (1 + growthRate);

    }

    public static double forecastMemo(double presentValue, double growthRate, int years, double[] memo) {

        if (years == 0) {

            return presentValue;

        }

        if (memo[years] != 0.0) {

            return memo[years];

        }

        memo[years] = forecastMemo(presentValue, growthRate, years - 1, memo) \* (1 + growthRate);

        return memo[years];

    }

    public static void main(String[] args) {

        double initialInvestment = 10000.0;

        double annualRate = 0.07;

        int years = 5;

        System.out.println("Using Basic Recursion:");

        double result1 = forecastValue(initialInvestment, annualRate, years);

        System.out.printf("Projected Value after %d years: %.2f%n", years, result1);

        System.out.println("\nUsing Memoization:");

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

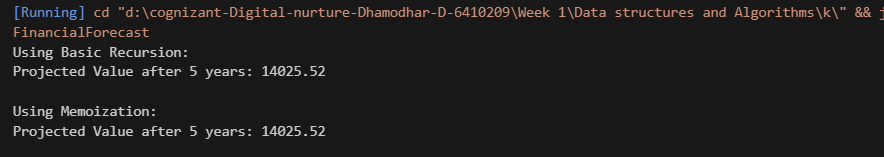
        double result2 = forecastMemo(initialInvestment, annualRate, years, memo);

        System.out.printf("Projected Value after %d years: %.2f%n", years, result2);

    }

}

**Output:**



**Analysis:**

**Time Complexity:**

Basic Recursive Method: O(n), where n is the number of years.

Memoized Version: Still O(n), but avoids redundant calculations.

**Optimization:**

Use **memoization** or convert recursion to an **iterative loop** to reduce stack overhead and improve performance in larger datasets.