**Exercise 7: Financial Forecasting**

**Code:**

**FinancialForecast.java**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class FinancialForecast {

static Map<Integer, Double> memo = new HashMap<>();

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

if (years == 0)

return initialAmount;

if (memo.containsKey(years))

return memo.get(years);

double result = (1 + growthRate) \* forecastValue(initialAmount, growthRate, years - 1);

memo.put(years, result);

return result;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter initial amount: ");

double initialAmount = sc.nextDouble();

System.out.print("Enter annual growth rate (in %): ");

double rate = sc.nextDouble();

double growthRate = rate / 100.0;

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

int years = sc.nextInt();

double futureValue = forecastValue(initialAmount, growthRate, years);

System.out.printf("\nForecasted value after %d years: ₹%.2f\n", years, futureValue);

System.out.println("\n--- Analysis ---");

System.out.println("Recursive approach time complexity: O(n)");

System.out.println("Without memoization, it would be exponential: O(2^n)");

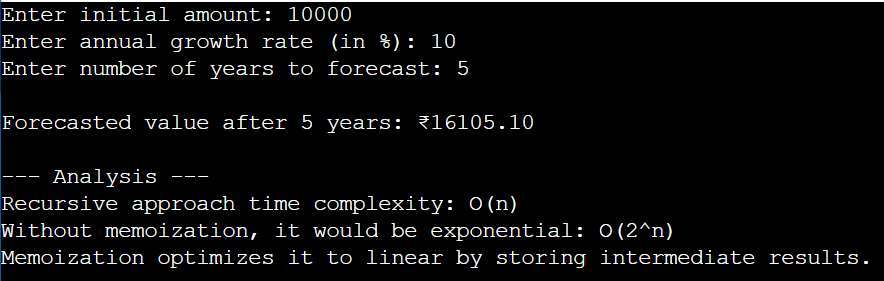
System.out.println("Memoization optimizes it to linear by storing intermediate results.");

sc.close();

}

}

**Output:**

****