**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Java code :**

public class FinancialForecasting {

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 predictMemoized(double currentValue, double growthRate, int years, double[] memo) {

if (years == 0) return currentValue;

if (memo[years] != 0) return memo[years];

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

}

public static void main(String[] args) {

double initialInvestment = 10000;

double annualGrowthRate = 0.1;

int futureYears = 5;

double futureValue = predictFutureValue(initialInvestment, annualGrowthRate, futureYears);

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

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

double futureValueOptimized = predictMemoized(initialInvestment, annualGrowthRate, futureYears, memo);

System.out.printf(" Predicted value after %d years (memoized): %.2f\n", futureYears, futureValueOptimized);

System.out.println("\n Time Complexity:");

System.out.println("Recursive: O(n) - one call per year");

System.out.println("Memoized: O(n) - avoids redundant calculations");

}

}

**Output :**

