**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

# THE CODE:

using System;

using System.Collections.Generic;

public class FinancialForecaster

{

public static double PredictFutureValue(double presentValue, double growthRate, int years)

{

if (years == 0)

return presentValue;

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

}

public static double PredictWithMemo(double presentValue, double growthRate, int years, Dictionary<int, double> memo = null)

{

memo ??= new Dictionary<int, double>();

if (years == 0)

return presentValue;

if (memo.ContainsKey(years))

return memo[years];

double result = PredictWithMemo(presentValue, growthRate, years - 1, memo) \* (1 + growthRate);

memo[years] = result;

return result;

}

}

class Program

{

static void Main(string[] args)

{

double presentValue = 1000;

double growthRate = 0.07;

int years = 10;

Console.WriteLine("Recursive Forecast:");

double futureValueRecursive = FinancialForecaster.PredictFutureValue(presentValue, growthRate, years);

Console.WriteLine($"Future Value (recursive): ${futureValueRecursive:F2}");

Console.WriteLine("\n Optimized Forecast with Memoization:");

double futureValueMemo = FinancialForecaster.PredictWithMemo(presentValue, growthRate, years);

Console.WriteLine($"Future Value (memoized): ${futureValueMemo:F2}");

}

}

# THE OUTPUT

