Financial Forecasting

```
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace FinancialForecasting
{
   public class FinancialForecaster
   {

      public decimal CalculateFutureValueRecursive(decimal currentValue, decimal
annualGrowthRate, int years)
      {

         if (years <= 0)
         {
            return currentValue;
         }


         decimal nextValue = currentValue * (1 + annualGrowthRate / 100m);
         return CalculateFutureValueRecursive(nextValue, annualGrowthRate, years - 1);
      }

      private Dictionary<(decimal, decimal, int), decimal> memoCache = new Dictionary<(decimal,
decimal, int), decimal>();

      public decimal CalculateFutureValueMemoized(decimal currentValue, decimal
annualGrowthRate, int years)
      {

         var key = (currentValue, annualGrowthRate, years);
```

```
        if (memoCache.TryGetValue(key, out decimal cachedValue))
        {
            return cachedValue;
        }

        if (years <= 0)
        {
            memoCache[key] = currentValue;
            return currentValue;
        }

        decimal nextValue = currentValue * (1 + annualGrowthRate / 100m);
        decimal result = CalculateFutureValueMemoized(nextValue, annualGrowthRate, years - 1);
        memoCache[key] = result;
        return result;
    }

    public decimal CalculateFutureValueIterative(decimal currentValue, decimal annualGrowthRate,
int years)
    {
        decimal result = currentValue;
        for (int i = 0; i < years; i++)
        {
            result *= (1 + annualGrowthRate / 100m);
        }
        return result;
    }
}

class Program
{
```

```
    static void Main(string[] args)
    {
        FinancialForecaster forecaster = new FinancialForecaster();

        decimal initialValue = 10000m;
        decimal growthRate = 5m;
        int years = 10;

        var stopwatch = Stopwatch.StartNew();
        decimal recursiveResult = forecaster.CalculateFutureValueRecursive(initialValue, growthRate,
years);
        stopwatch.Stop();
        Console.WriteLine($"Recursive Result: {recursiveResult:C2}");
        Console.WriteLine($"Time taken: {stopwatch.ElapsedTicks} ticks");

        stopwatch.Restart();
        decimal memoizedResult = forecaster.CalculateFutureValueMemoized(initialValue,
growthRate, years);
        stopwatch.Stop();
        Console.WriteLine($"\nMemoized Result: {memoizedResult:C2}");
        Console.WriteLine($"Time taken (first run): {stopwatch.ElapsedTicks} ticks");

        stopwatch.Restart();
        decimal iterativeResult = forecaster.CalculateFutureValueIterative(initialValue, growthRate,
years);
        stopwatch.Stop();
        Console.WriteLine($"\nIterative Result: {iterativeResult:C2}");
        Console.WriteLine($"Time taken: {stopwatch.ElapsedTicks} ticks");

        Console.WriteLine("\nTesting memoization with repeated calculations...");
        stopwatch.Restart();
        for (int i = 0; i < 1000; i++)
```

```
        {
            forecaster.CalculateFutureValueMemoized(initialValue, growthRate, years);
        }
        stopwatch.Stop();
        Console.WriteLine($"1000 memoized calls: {stopwatch.ElapsedTicks} ticks");

        stopwatch.Restart();
        for (int i = 0; i < 1000; i++)
        {
            forecaster.CalculateFutureValueRecursive(initialValue, growthRate, years);
        }
        stopwatch.Stop();
        Console.WriteLine($"1000 recursive calls: {stopwatch.ElapsedTicks} ticks");
    }
  }
}
```