**Financial Forecasting**

**1. Understand Recursive Algorithms**

* **Recursion** is a programming technique where a method calls itself to solve smaller instances of the same problem.
* It's useful when:
  + A problem can be divided into sub-problems of the same type.
  + Iterative solutions become complex or harder to read.
* In financial forecasting, recursion can help calculate future values based on a repeated pattern (e.g., annual growth).

**2. Setup**

V(n) = FV(n-1) \* (1 + growthRate)

Where:

* FV(n) is the future value after n years
* growthRate is the annual percentage increase

**3. Implementation**

using System;

namespace FinancialForecasting

{

    class Program

    {

        static double CalculateFutureValue(double currentValue, double growthRate, int years)

        {

            if (years == 0)

                return currentValue;

            return CalculateFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);

        }

        static void Main(string[] args)

        {

            Console.WriteLine("Enter current value:");

            double currentValue = Convert.ToDouble(Console.ReadLine());

            Console.WriteLine("Enter annual growth rate (e.g., 0.05 for 5%):");

            double growthRate = Convert.ToDouble(Console.ReadLine());

            Console.WriteLine("Enter number of years:");

            int years = Convert.ToInt32(Console.ReadLine());

            double futureValue = CalculateFutureValue(currentValue, growthRate, years);

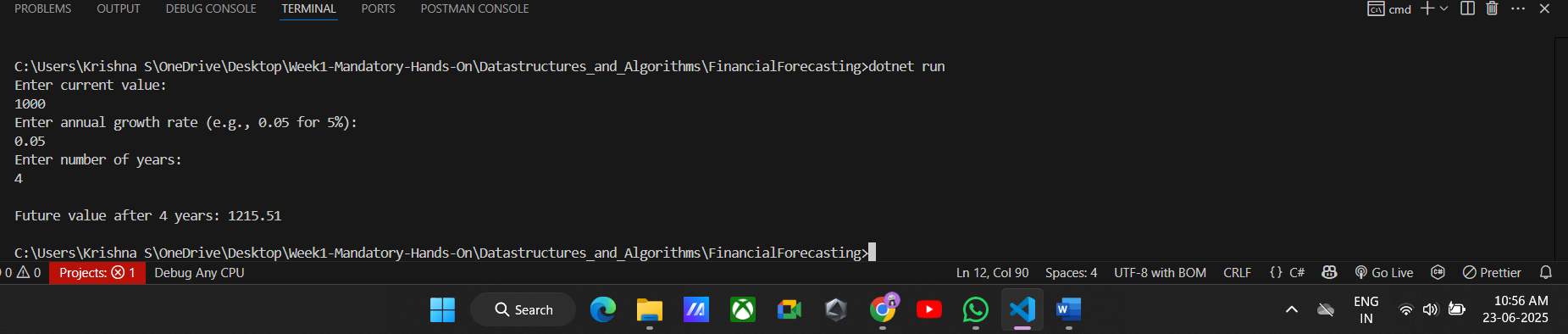
            Console.WriteLine($"\nFuture value after {years} years: {futureValue:F2}");

        }

    }

}

**4. Output**

****

**5. Analysis**

**Time Complexity:**

* Each recursive call reduces years by 1.
* So for n years, we have **O(n)** time complexity.
* Space complexity is also **O(n)** due to function call stack.

**Optimization:**

**Use Iteration** instead of Recursion:

* Avoids overhead of multiple function calls.
* More efficient for large n.

double value = currentValue;

for (int i = 0; i < years; i++)

{

    value \*= (1 + growthRate);

}

The iterative method maintains O(n) time complexity but reduces space complexity to O(1), making it more efficient for large inputs.