**Exercise 7: Financial Forecasting**

## Problem Statement:

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

**RECURSION:**

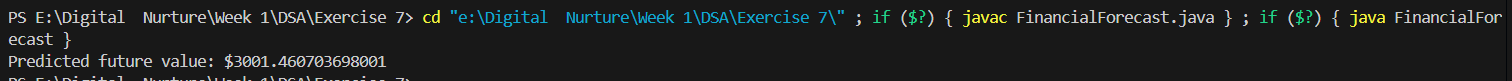
**Recursion** is when a function **calls itself** to solve smaller versions of the same problem.  
It’s useful when a problem can be broken down into smaller sub-problems that follow the same pattern.

In financial forecasting, recursion can be used to calculate future values based on past trends or growth rates.

**IMPLEMENTATION:**

public class FinancialForecast {  
  
 public static double forecast(double currentValue, double growthRate, int years) {  
 if (years == 0) {  
 return currentValue;  
 }  
 return forecast(currentValue \* (1 + growthRate), growthRate, years - 1);  
 }  
  
 public static void main(String[] args) {  
 double futureValue = forecast(2000, 0.07, 6);  
 System.out.println("Predicted future value: $" + futureValue);  
 }  
}

**OUTPUT:**

****

**ANALYSIS:**

### Time Complexity:

* The time complexity is **O(n)** because the function calls itself **once per year**.
* So, for 6 years, 6 recursive calls are made.

**Optimization:**

 Right now, it’s fine for small values like 5 or 10 years.

 But for large values, we can **convert recursion to iteration** to avoid stack overflow.

**CODE:**

public static double forecastIterative(double currentValue, double growthRate, int years) {  
 for (int i = 0; i < years; i++) {  
 currentValue \*= (1 + growthRate);  
 }  
 return currentValue;  
}