# **Assignment: Exercise 7 – Financial Forecasting**

## **1. Understanding Recursive Algorithms**

In this assignment, I worked on recursion, which is when a method calls itself to solve a smaller version of the same problem. It’s like solving a big problem step-by-step by breaking it down into smaller ones until we hit a base case.

I’ve seen recursion used in problems like factorials and Fibonacci series. In this task, I used recursion to predict financial growth year by year based on a fixed growth rate. Recursion makes logic look cleaner sometimes, but it’s not always the best for performance unless we handle it properly.

## **2. Setup: Creating a Recursive Forecast Method**

I wrote a method that calculates the future value using a recursive approach. It takes the present value, growth rate, and number of years and returns the forecasted value.

The function keeps calling itself with one year less until it reaches 0 years, which is the base case.

## **3. Implementation: Recursive Forecast Function**

public class FinancialForecast {

public static double forecastValue(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

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

}

public static void main(String[] args) {

double presentValue = 1000;

double growthRate = 0.1; // 10%

int years = 5;

double futureValue = forecastValue(presentValue, growthRate, years);

System.out.println("Forecasted value after " + years + " years: " + Math.round(futureValue \* 100.0) / 100.0);

}

}

This method calculates the future value recursively, and I tested it with some sample inputs like 1000, 10% growth, and 5 years.

## **4. Analysis: Time Complexity and Optimization**

### **Time Complexity:**

The time complexity of this recursive method is **O(n)** where n is the number of years. This is because the function is called once for each year until we reach year 0.

### **Optimization:**

Even though recursion looks neat, it’s not the best for large inputs. It may cause stack overflow or just become slower. We can optimize it by:

* Using **memoization** to store already calculated values so we don’t do the same work again.
* Or simply use a **loop (iteration)** which is more efficient in this case.

## **5. Output**

Forecasted value after 5 years: 1610.51

This output shows that the present value of 1000 grows to about **1610.51** after 5 years with a 10% yearly growth. So the recursive method works correctly.