**Exercise 7: Financial Forecasting**

**Scenario:**

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

**What is Recursion?**

Recursion is a programming technique where a method calls itself to solve a problem in smaller parts.

**Why Use Recursion?**

* Useful for problems that are naturally repetitive or hierarchical.
* Makes code more readable for problems like factorials, Fibonacci series, or growth-based forecasting.

**Setup: Define the Forecast Problem**

Let’s say:

* We are given an initial value (e.g., current revenue).
* We have a fixed annual growth rate (e.g., 10%).
* We need to predict the future value after n years using:

FutureValue(n)=FutureValue(n−1)×(1+growthRate)

**Analysis**

**Time Complexity:**

* **Time Complexity**: O(n)  
  Each recursive call reduces the problem size by 1 until it hits the base case.

**Space Complexity:**

* **Space Complexity**: O(n)  
  Each call adds a frame to the call stack, leading to potential stack overflow for very large n.

**Optimizing the Recursive Solution**

**Convert to Iterative:**

You can use a loop to avoid deep recursion:

public static double predictFutureValueIterative(double initialValue, double growthRate, int years) {

double result = initialValue;

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

result \*= (1 + growthRate);

}

return result;

}

**Use Memoization (optional for complex cases):**

If future values depend on overlapping subproblems (like Fibonacci), memoization helps. But for this problem, it's not needed since each subproblem is unique.