**WEEK1 – Data Structures and Alogorithms HandsOn**

# Exercise 7: Financial Forecasting 1. Understand Recursive Algorithms

Recursion is a programming concept where a method calls itself to solve smaller parts of a problem. It is useful for problems that have repetitive structure. In financial forecasting, we calculate values year-by-year recursively.

# Setup

**Goal**: Create a method to calculate future value based on:

* Starting value
* Annual growth rate
* Number of years

**Formula**: futureValue(n) = (1 + rate) \* futureValue(n - 1), with base case: futureValue(0) =

current value

# Implementation

public class FinancialForecast {

public static double futureValue(int years, double currentValue, double growthRate) { if (years == 0) { return currentValue;

}

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

}

public static void main(String[] args) { double currentValue = 1000.0; double growthRate = 0.05;

int years = 10;

double result = futureValue(years, currentValue, growthRate); System.out.printf("Future Value after %d years = Rs.%.2f\n", years, result);

} }

# Analysis

* Time Complexity: O(n), one recursive call per year.
* Space Complexity: O(n), due to recursion stack.

Problem: High space usage for large n.

Optimization: A. Iterative:

for (int i = 0; i < years; i++) { currentValue \*= (1 + growthRate);

}

B. Using Math.pow: return currentValue \* Math.pow(1 + growthRate, years);

Summary

| Approach | Time | Space | Use Case |

|-----------|------|-------|----------|

| Recursive | O(n) | O(n) | Simple understanding |

| Iterative | O(n) | O(1) | Medium input size |

| Optimized | O(1) | O(1) | Best for large inputs |



