**Week 1 - Algorithms and Data Structures - Hands-On Exercise**

**Student Name: Jitesh Sen  
Email:** [**jiteshsen2002@gmail.com**](mailto:jiteshsen2002@gmail.com) **Superset ID: 6387362**

**Exercise 7: Financial Forecasting**

**1. Understanding Recursive Algorithms**

**What is Recursion?**

In simple words, Recursion is a method that calls itself. The concept of recursion involves breaking down a problem into smaller subproblems and solving each subproblem by calling itself. It needs two parts: a base case (when to stop) and a recursive case (the method calling itself with different parameters).

Recursion simplifies financial problems because growth naturally builds on previous values. Each year's investment grows from the last year, which is exactly how recursion works - solving the current problem using the solution from the previous step.

**2. Setup**

I created a method to calculate future value using recursion where each year's value depends on applying the growth rate to the previous year's value.

**3. Implementation**

I implemented a recursive algorithm that calculates future investment values by applying the growth rate year by year. The method takes the current value, growth rate, and remaining years, then either returns the final value (base case) or calculates the next year's value and calls itself for the remaining years.

**4. Analysis**

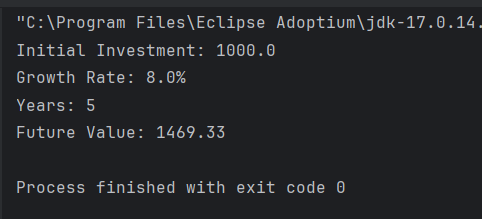
**Time Complexity:** The recursive algorithm has O(n) time complexity where n equals the number of years, because it makes one recursive call for each year.

**Optimization:** To avoid excessive computation and stack overflow issues, I can optimize by:

* Using an iterative approach instead of recursion (same result, no stack issues).
* Using the direct mathematical formula for compound growth (fastest option).
* These optimizations reduce space complexity from O(n) to O(1).

**Conclusion:** Recursion naturally fits financial forecasting but iterative solutions are more practical for real applications.

**Program Output Recursion:**



**Program Output Optimized:**

