# Exercise 7: Financial Forecasting

**1. Understanding Recursive Algorithms**

**Recursion** is a programming technique where a function calls itself to solve a smaller instance of the same problem. It is particularly useful for problems that can be broken down into similar subproblems, such as computing factorials, Fibonacci numbers, or in this case, future financial value over a period of time.

**Benefits of Recursion:**

* Simplifies code by eliminating loops in problems with repetitive patterns.
* Makes code easier to understand for divide-and-conquer problems.
* Useful when each result depends on the previous result (as in compounding).

**2. Analysis**

**Time Complexity:**

* **O(n)** — where n is the number of years.
* One recursive call is made for each year.

**Space Complexity:**

* **O(n)** — due to the recursive call stack, each function call adds a frame until it reaches the base case.

**How to optimize the recursive solution to avoid excessive computation**

* Recursion may cause **stack overflow** for large n. In this case, iteration is more memory-efficient.
* **Use Memoization or Dynamic Programming**  
  Store results of previous calculations so you don’t repeat the same work again. This saves time and makes the program run faster.
* For performance-critical systems, use **iterative methods** instead of recursion.