**Exercise 7: Financial Forecasting**

1)Understanding Recursion

Recursion is a problem-solving technique where a function calls itself to solve smaller instances of the same problem. It's like breaking down a complex problem into smaller, more manageable subproblems until you reach a simple base case that can be solved directly.

2)Essential Components:

Base Case: This is the simplest form of the problem that can be solved without further recursion. It's the stopping condition to prevent infinite loops.

Recursive Case: This is where the function calls itself with a modified input, bringing the problem closer to the base case.

3)Advantages of Recursion

Simplicity: Recursive solutions often mirror the problem's structure, leading to elegant and concise code.

Divide and Conquer: Many problems can be efficiently solved by breaking them down into smaller subproblems, a strategy naturally suited to recursion.

Natural for Certain Problems: Problems involving trees, graphs, and mathematical sequences often have recursive patterns.

4)Time Complexity

The time complexity of a recursive function depends on the problem and the implementation. It's often expressed using recurrence relations. Analyzing these relations can help determine the overall time complexity.

Example: For the factorial function, the time complexity is O(n), where n is the input number.

5)Optimization Techniques

Memoization: Store the results of function calls to avoid redundant calculations. This is particularly useful for overlapping subproblems.

Tail Recursion: In some languages, tail-recursive functions can be optimized by the compiler to iterative code, improving performance.

Iterative Equivalent: If possible, convert the recursive function into an iterative one to potentially eliminate function call overhead.