**1. Explain the concept of recursion and how it can simplify certain problems.**

Recursion is a technique where a function repeatedly calls itself to solve a task by dividing it into smaller, manageable subproblems. This continues until it reaches a defined stopping condition known as the base case.  
• Recursion often results in cleaner and easier-to-understand code.  
• It is particularly useful in solving problems involving repetition, patterns, or hierarchical structures such as tree traversal, calculating factorials, generating Fibonacci numbers, or computing compound interest.  
• In financial forecasting, recursion helps in easily calculating future values year by year, avoiding the need for complex loop structures.

**4(a) Discuss the time complexity of your recursive algorithm.**

The recursive financial forecasting method calculates future values by calling itself once for every year, multiplying the previous result by the growth rate.  
Time Complexity:  
• **Time Complexity:** O(n) — because it performs one recursive call per year.  
• **Space Complexity:** O(n) — each call uses stack memory, which grows with the number of years.  
While this is effective for small values of *n*, larger values can cause performance issues due to increased recursion depth.

**4(b) Explain how to optimize the recursive solution to avoid excessive computation.**

To make the recursive approach more efficient and avoid problems like stack overflow in deep recursion, we can use two strategies:  
**i. Use Iteration Instead of Recursion**  
• Replace the recursive logic with a loop to avoid using the call stack.  
• This improves space efficiency by reducing space complexity to O(1).  
**ii. Memoization (for overlapping subproblems)**  
• Store the results of previous calculations in a structure like an array or map.  
• Though not necessary for this linear problem, it's highly effective in cases like Fibonacci where results repeat.