1. **Understand Recursive Algorithms:**

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

Solution:

Recursion is when a method calls itself to solve a smaller subproblem until it reaches a base case.

Example:

int factorial(int n) {

if (n == 1) return 1;

return n \* factorial(n - 1);

}

1. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
2. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth .

Solution:

Implemented as code.

1. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

Solution:

**Time Complexity**

* The recursive function calls itself once per year.
* So, **Time Complexity**: O(n) where n = years
* **Space Complexity**: Also O(n) due to the call stack.

To optimize a recursive solution and avoid excessive computation, use **memoization** to store previously computed results or switch to **iteration** to eliminate the overhead of recursive calls.