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

Recursion is a programming technique in which a function calls itself to solve a smaller instance of the same problem.

Every recursive function has:

* A base case — the condition under which the function stops calling itself.
* A recursive case — where the function keeps calling itself with a smaller input.

How Recursion Simplifies Problems

* Recursion is useful for problems that can be broken down into smaller, similar subproblems.
* It leads to cleaner and shorter code for tasks like:
* Calculating factorials
* Computing Fibonacci numbers
* Traversing trees or graphs
* Solving problems with repeated patterns (such as in backtracking)

**Discuss the time complexity of your recursive algorithm.**

* The recursive method calculateFutureValue calls itself once for each year.
* For n years, the function is called n + 1 times (including the base case).
* Therefore, the time complexity is O(n).
* This is considered efficient for small to moderate values of n, such as forecasting over a few decades.
* However, recursion introduces function call overhead due to repeated stack usage, which can be a drawback for very large n.

**Explain how to optimize the recursive solution to avoid excessive computation.**

* Ways to optimize the recursive solution:
* Use iteration instead of recursion to reduce function call overhead.
* Tail recursion helps in some languages, but Java does not optimize tail calls.
* Apply memoization in complex cases to avoid recalculating the same results.
* For large values of n, prefer an iterative or formula-based approach to avoid deep recursion.