Q1. Explain the concept of recursion and how it can simplify certain problems.

Ans.

What is recursion?

* **Recursion** is when a function calls itself to solve smaller instances of the same problem.
* It’s useful for problems that naturally break into smaller subproblems (e.g., calculating factorial, Fibonacci numbers, etc.).

Why recursion is used?

* Simplifies code for repetitive calculations.
* Eliminates the need for complex loops in problems that can be expressed mathematically.

Q2. Discuss the time complexity of your recursive algorithm.

Ans.

Time Complexity:

* **T(n) = T(n - 1) + O(1)**
* So, **Time Complexity = O(n)** — one recursive call per year.

Space complexity:

* **O(n)** — each recursive call adds to the call stack.

Q3. Explain how to optimize the recursive solution to avoid excessive computation.

* **Memoization** (Storing results)
* **Iterative Approach** — best for performance and avoiding stack overflow.

Optimized code:

public static double futureValueIterative(double currentValue, double rate, int years) {

double result = currentValue;

for (int i = 0; i < years; i++) {

result \*= (1 + rate);

}

return result;

}