Answers

1. **Recursion** is a programming technique where a method calls itself in order to solve a problem. Recursive algorithms can simplify complex problems by breaking them down into smaller, more manageable subproblems. Each recursive call typically has a base case that stops the recursion, preventing it from running indefinitely.

Recursion can be particularly useful in problems where the solution involves solving the same problem multiple times with smaller inputs, such as in mathematical computations, tree traversals, and dynamic programming.

4. **Analysis**

**Time Complexity**

The time complexity of this recursive algorithm is O(n), where nnn is the number of periods. This is because the function makes a single recursive call for each period until it reaches the base case (when periods equals zero).

**Optimization**

One common issue with recursion is **excessive computation** due to repeated calculations of the same values. In this specific case, recursion does not incur this overhead because each call operates on distinct values. However, for other problems, such as those that exhibit overlapping subproblems (e.g., Fibonacci sequence), optimizing through **memoization** or **dynamic programming** can help.