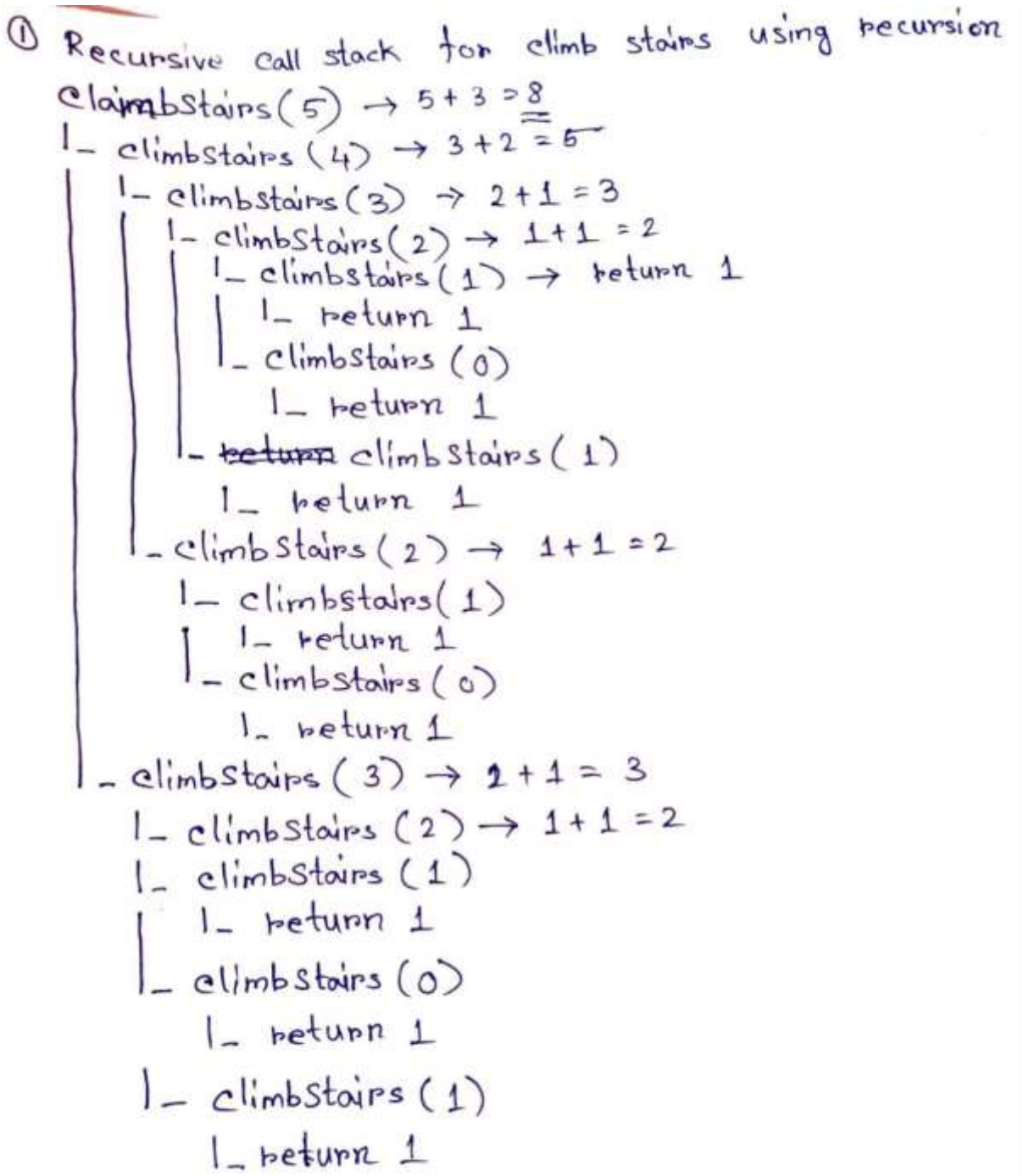
Climb Stairs LeetCode

This C++ program calculates the number of distinct ways to climb a given number of stairs using two different approaches: a naive recursive approach and a more optimized approach with memoization.

Recursive function to calculate the number of distinct ways to climb the stairs

- The function uses recursion to explore all possible paths to climb the stairs. The base cases are when **nthStair** is less than 0 (no ways to climb) or when **nthStair** is 0 or 1 (only one way to climb).
- The recursive case calculates the number of ways to climb **nthStair** by summing the ways to climb **(nthStair - 1)** and **(nthStair - 2)**, as you can take one or two steps at a time.
- **Time Complexity: This approach has exponential time complexity since it explores many overlapping subproblems, resulting in redundant calculations O(2^n)**
- **Space Complexity: In the recursive approach (climbStairs), the space complexity is primarily determined by the depth of the recursion, which can be significant for larger nthStair values. It has exponential space complexity O(n) in the worst case due to the recursive call stack.**

Recursive call tree for this approach:



Recursive function to calculate the number of distinct ways to climb the stairs with memorization

- The memoization map **memo** is used to avoid redundant calculations and store the number of ways to climb stairs for each **nthStair**.

- Before performing a recursive call, the function first checks if the number of ways to climb the **nthStair** is already calculated and available in **memo**. If so, it directly returns the stored result.
- If the result is not present in **memo**, it calculates the number of ways recursively and stores the result in **memo** for future reference.
- **Time Complexity: This approach has a time complexity of O(n) as each stair is calculated only once, and the space complexity is O(n) due to the storage of results in the memo map.**
- **Space Complexity: In the memoization approach (climbStairsMemoization), the space complexity is improved to O(n) due to the usage of the memo unordered map. The map stores the result for each nthStair, reducing redundant calculations and optimizing the overall space usage.**

Recursive call tree for the approach: