In this assignment I wrote code to solve the Towers of Hanoi puzzle using both stacks and recursion with any number of disks.

Stack
The code for the stack solution was easier for me to find, but longer to write. It required me to find the gameplay pattern (I saw the pattern being: if it's an even number of disks then make the legal move between pegs AC, then AB, then BC, and repeat until solved. If the number of disks is odd then make the legal move between pegs AB, then AC, then BC, and repeat until solved.). This required the use of many if else statements and some 2 while loops. That was the most complex part of the stack solution, using the stacks themselves was not very difficult once I figured out how they work and how I could use the helper functions in stack.h to my advantage.

Recursion
The recursive solution was more difficult for me to see and took a while to find but once I wrapped my head around the building from the base case up I found the solution and it was simple to implement. It's only a few lines of code but in my opinion, it is more of a complex solution because everything isn’t right out in front of you. It's like having to dive to the bottom of the rabbit hole and back up to get the solution compared to the iterative solution which is just following a path around and around until you’re done.

Difference between stack and recursion
Stack uses actual memory manipulation and solves the problem like you would physically: checking what moves are legal, how many total disks you have, etc. and makes if else decisions in a loop to come to the solution. Recursion breaks down the whole problem into a smaller problem, then that smaller problem into an even smaller problem and so on until the problem cannot be any smaller. Then it solves the smallest problem and returns that solution up one level and then that level can be solved and so on until all the problems are solved, thereby solving the entire problem. In this case, the smallest (AKA easiest) problem for the Towers of Hanoi was if there was only one disk on a stack that needed to be moved, so that became the base case. From there we broke down the problem by alternating between the auxiliary peg and the final peg, pausing each problem and going down deeper into the recursion. What makes them unique is that the stack solution is iterative, while the recursive solution is, well, recursive and breaks down a problem into smaller instances of the same problem.