Analysis of restricted tower of hanoi:

in this code we used the recursive algorithm as it is specifically designed for the restricted tower of Hanoi ptoblem where every single move must involve peg b. we chose this approach as it mirrors similar structure of the problem and gives a elegant mathematical answer that could be easily accepted

recursive method directly carrys out the constraint that moves between peg a and peg c are not allowed this requires all of the disk transfers to pass through peg b which creates a predictable pattern that could be shown through recurrence relations

there could have been multiple alternative algorithms such as :

1- The standard tower of Hanoi algprithm : this uses a 3step recursive process but it allows direct moving from a to c which makes it not suitable for this task
2- The Iterative approach: this approach uses stack based state tracking with certain movement rules but it would be harder to implement while following the peg b constraint
3- BFS (breadth-first search): this would show the problem as a state space search and would explore all the possible cinfigrations but this would be not suitable as the number of game states increses a lot making it impossible to run larger inputs as it would exceed the memory limit

How our code works ?

The program consists of two core functions and a main the moves needed function calculates the number of moves required for n disks using the formula 3^n-1 and it uses an iterative loop to compute the power,making surer of accuracy and its effeicency

The moveDisks function is the recursive part of this solution. For a single disk it performs two movements: from the source peg to auxiliary peg (B), and then from B to the destination, strictly following the rule. For n disks, it outputs a sophisticated five-step sequence: it first moves the top n-1 disks from the source to the destination *using the auxiliary peg*; it then moves the largest disk from the source to the auxiliary; it moves the n-1 disks from the destination back to the source; it moves the largest disk from the auxiliary to the final destination; and finally, it moves the n-1 disks from the source to the destination again. A reference parameter is used to meticulously count every move throughout this recursive process.

Complexity analysis :

The algorithm has exponential time complexity of O(3^n) because each recursive call generates three smaller subproblems. The space complexity is O(n) due to the recursion stack depth. The recursive approach differs from iterative in that it naturally breaks down the problem into smaller subproblems and uses the call stack to manage state, whereas an iterative approach would require explicit stack management and more complex state tracking while maintaining the peg B constraint.