Delete Middle Element of a Stack

Given a stack with push(), pop(), empty() operations, delete the middle of the stack without using any additional data structure.

Middle: ceil((size_of_stack+1)/2) (1-based index)

Input:	Output:
Stack = {1, 2, 3, 4, 5}	ModifiedStack = {1, 2, 4, 5}

Approach:

If the stack has N elements, then we need to somehow store N/2 elements of the stack and then delete (N/2 + 1)-th element of the stack. Then we can push the top N/2 elements that we have stored. To store N/2 elements we have two methods: either we can go for an iterative approach and use a temporary stack or we can go for a recursive approach and use a recursive stack.

Algorithm for Iterative approach:

1. Create an empty temporary stack temp.
2. Pop N/2 elements from the given stack and Push it into the temp stack.
3. Pop top of the given stack , this is the targeted middle element.
4. Pop all elements from the temp stack and Push it into the given stack .
5. This will be the final stack after deleting the middle element.

Algorithm for Recursive Approach:

1. Create a recursive function which will accept the current stack and initial size of the input stack.
2. In each call we first check if N/2 elements have already been deleted , then the top of stack will be the targeted middle element, we will pop it and stop the recursive call
3. Else we pop the top of stack and store it into a variable x, and do the recursive call for the current stack after popping.
4. Once the recursive call completes, push x into the stack .
5. The stack obtained at the end of the call is the final required stack.

Time Complexity: O(N)
Space Complexity: O(N)

CODE:

```
class Solution
{
    public:
  //  Function to delete middle element of a stack.
    void deleteInStack(stack<int>&s,int sizeOfStack)
    {
        if(s.size() == ceil((sizeOfStack+1)/2))
        {
            s.pop();
            return;
        }
        int x = s.top();
        s.pop();
        deleteInStack(s,sizeOfStack);
        s.push(x);
    }

    void deleteMid(stack<int>&s, int sizeOfStack)
    {
        deleteInStack(s,sizeOfStack);
    }
};
```