Reverse a Stack Using Recursion

You are given a stack of integers, and your task is to write a function that reverses the stack using recursion. You are not allowed to use any additional data structure (like arrays, lists, or another stack). The only operations you are allowed to perform are push, pop, and peek on the stack. The reversal must be done using recursion only.

Input:
- A stack of integers. The stack may contain positive, negative, or zero values.

Output:
- The stack should be reversed, meaning the element that was at the bottom of the original stack should become the topmost element after the reversal.

Examples:
- Example 1
  Input: [1, 2, 3, 4]
  Output: [4, 3, 2, 1]
  Explanation:
  - Initially, the stack contains [1, 2, 3, 4].
  - After the reversal using recursion, the stack becomes [4, 3, 2, 1].

Constraints:
- You are not allowed to use any additional data structure like arrays or lists.
- You can only use stack operations (push, pop, peek) and recursion.
- The input stack can have up to 10^4 elements.

Test Cases:
1. Input: [3, 2, 1]
   Output: [1, 2, 3]
2. Input: [5]
   Output: [5]
3. Input: [-5, -10, -15]
   Output: [-15, -10, -5]
4. Input: []

Output: []

5. Input: [3, 3, 3]
   Output: [3, 3, 3]

Edge Cases:

1. Single Element: If the stack contains only one element, it remains the same after reversal.
2. Empty Stack: If the stack is empty, no operation is needed.
3. All Elements are the Same: If all elements in the stack are the same, the order will not change after reversal.
4. Negative Numbers: The stack may contain negative integers, and they should still be reversed correctly.