Task 6: Searching for a Sequence in a Stack. Given a stack and a smaller array representing a sequence, write a function that determines if the sequence is present in the stack. Consider the sequence present if, upon popping the elements, all elements of the array appear consecutively in the stack.

Code Implementation

```
import java.util.Stack;
public class StackSequenceChecker {
  public static boolean isSequencePresent(Stack<Integer> stack, int[] sequence) {
    // Reverse the sequence array to facilitate checking from end to start
    for (int i = 0; i < sequence.length / 2; i++) {
      int temp = sequence[i];
      sequence[i] = sequence[sequence.length - 1 - i];
      sequence[sequence.length - 1 - i] = temp;       }
    Stack<Integer> tempStack = new Stack<>();
    for (int num : sequence) {
      tempStack.push(num);       }
    while (!stack.isEmpty() && !tempStack.isEmpty()) {
      if (stack.pop().equals(tempStack.peek())) {
        tempStack.pop();          }       }
    return tempStack.isEmpty();    }
  public static void main(String[] args) {
    Stack<Integer> stack = new Stack<>();
    stack.push(3);
    stack.push(1);
    stack.push(4);
    stack.push(2);
    stack.push(5);
    int[] sequence = {4, 2, 5};
    boolean result = isSequencePresent(stack, sequence);
    System.out.println("Is the sequence present? " + result);    }}
```

Explanation of the Code

1. **Reverse the Sequence:**
   - We reverse the sequence array because stacks are LIFO (Last In, First Out), and we want to check the sequence from the end to the start as we pop elements from the stack.
2. **Temporary Stack:**
   - We use a temporary stack tempStack to store the reversed sequence. This helps in checking the sequence from the end as we pop elements from the original stack.
3. **Check Sequence in Stack:**
   - We traverse the original stack, popping elements one by one.
   - If the popped element matches the top of tempStack (the current element in the sequence we are checking), we pop the top element from tempStack.
   - If all elements of the sequence are found in order (i.e., tempStack becomes empty), the sequence is present in the stack.
4. **Result:**
   - The function returns true if tempStack is empty at the end, meaning the entire sequence was found in order in the stack. Otherwise, it returns false.
5. **Main Method:**
   - Demonstrates the function with a sample stack and sequence.
   - Prints the result of whether the sequence is present in the stack.