1209. Remove All Adjacent Duplicates in String II

You are given a string `s` and an integer `k`, a `k` **duplicate removal** consists of choosing `k` adjacent and equal letters from `s` and removing them, causing the left and the right side of the deleted substring to concatenate together.

We repeatedly make `k` **duplicate removals** on `s` until we no longer can.

Return the final string after all such duplicate removals have been made. It is guaranteed that the answer is unique.

Example 1:

Input: s = "abcd", k = 2
Output: "abcd"
Explanation: There's nothing to delete.

Example 2:

Input: s = "deeedbbcccbdaa", k = 3
Output: "aa"
Explanation: First delete "eee" and "ccc", get "ddbbbdaa"
Then delete "bbb", get "dddaa"
Finally delete "ddd", get "aa"

Example 3:

Input: s = "pbbcggttciiippooaais", k = 2
Output: "ps"

```
def removeDuplicates(self, s: str, k: int) -> str:
        stack = []
        for i in range(len(s)):
            if i==0 or not len(stack):
                stack.append([s[i],1])
            else:
                if stack[-1][0]==s[i]:
                    temp = stack[-1][1]
                    if temp+1==k:
                        stack.pop()
```

```
                else:
                    _,count = stack.pop()
                    count = count+1
                    stack.append([s[i],count])
            else:
                stack.append((s[i],1))
        return ''.join([c*i for c,i in stack])
```