Candy Crush 5

- Write a function to crush candy in one dimensional board. In candy crushing games, groups of like items are removed from the board. In this problem, any sequence of 3 or more like items should be removed and any items adjacent to that sequence should now be considered adjacent to each other. This process should be repeated as many time as possible. You should greedily remove characters from left to right.

Input Format

- String s

Constraints

- Time:- 1Sec

Output Format

- String s

Sample Input 0

```
aaabbbc
```

Sample Output 0

```
c
```

Explanation 0

Explanation: 1. Remove 3 'a': "aaabbbbc" => "bbbbc" 2. Remove 4 'b': "bbbbc" => "c"

Sample Input 1

```
aabbbacd
```

Sample Output 1

```
cd
```

Explanation 1

Explanation: 1. Remove 3 'b': "aabbbacd" => "aaacd" 2. Remove 3 'a': "aaacd" => "cd"

More

Java 8

```java
import java.util.*;

public class CandyCrush {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
       // System.out.print("Enter candy board: ");
        String board = scanner.nextLine();
        String crushed = candyCrush(board);
       // System.out.println("Original board: " + board);
        System.out.println(crushed);
    }

    public static String candyCrush(String board) {
        Stack<int[]> stack = new Stack<>();
        int i = 0;
        int n = board.length();
        while (i < n) {
            int j = i;
            while (j < n && board.charAt(j) == board.charAt(i)) {
                j++;
            }
            if (j - i >= 3) {
                // found a sequence to crush
                stack.push(new int[]{i, j});
            }
            i = j;
        }

        if (stack.isEmpty()) {
            // no sequences found, return original board
            return board;
        }

        // create new board with sequences removed
        StringBuilder sb = new StringBuilder();
        int lastRemoved = -1;
        for (i = 0; i < n; i++) {
            if (!stack.isEmpty() && i == stack.peek()[0]) {
                int[] seq = stack.pop();
                lastRemoved = seq[1] - 1;
            } else if (lastRemoved >= i) {
                // skip over items adjacent to removed sequence
                continue;
            } else {
                sb.append(board.charAt(i));
            }
        }

        // recursively crush new board until no more sequences are found
        return candyCrush(sb.toString());
    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔ Testcase 1 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
aaabbbc
```

Your Output (stdout)

```
c
```

Expected Output

```
c
```