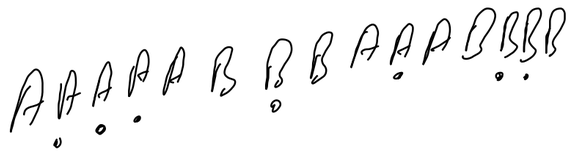

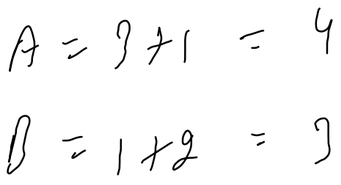

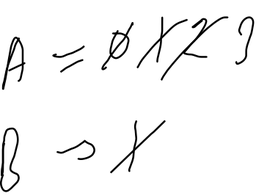

2038. Remove Colored Pieces if Both Neighbors are the Same Color

There are `n` pieces arranged in a line, and each piece is colored either by `'A'` or by `'B'`. You are given a string `colors` of length `n` where `colors[i]` is the color of the `i`th piece.

Alice and Bob are playing a game where they take **alternating turns** removing pieces from the line. In this game, Alice moves **first**.

- Alice is only allowed to remove a piece colored `'A'` if **both its neighbors** are also colored `'A'`. She is **not allowed** to remove pieces that are colored `'B'`.
- Bob is only allowed to remove a piece colored `'B'` if **both its neighbors** are also colored `'B'`. He is **not allowed** to remove pieces that are colored `'A'`.
- Alice and Bob **cannot** remove pieces from the edge of the line.
- If a player cannot make a move on their turn, that player **loses** and the other player **wins**.

Assuming Alice and Bob play optimally, return `true` *if Alice wins, or return* `false` *if Bob wins.*

```
Input: colors = "AAABABB"
Output: true
Explanation:
AAABABB -> AABABB
Alice moves first.
She removes the second 'A' from the left since that is the only 'A' whose neighbors are both 'A'.

Now it's Bob's turn.
Bob cannot make a move on his turn since there are no 'B's whose neighbors are both 'B'.
Thus, Alice wins, so return true.
```

```java
class Solution {
    public boolean winnerOfGame(String colors) {
        int[] countSteps = new int[2]; // 0th index for alice, 1st index for bob.

        int n = colors.length(), i = 0;
        while (i < n) {
            int countChar = 0;
            char ch = colors.charAt(i);
            while (i < n && ch == colors.charAt(i)){
                i++;
                countChar++;
            }

            countSteps[ch - 'A'] += Math.max(0,countChar - 2);
        }

        return countSteps[0] > countSteps[1];
    }
}
```