# 题目

给你一个由大小写英文字母组成的字符串 s 。

一个整理好的字符串中，两个相邻字符 s[i] 和 s[i+1]，其中 0<= i <= s.length-2 ，要满足如下条件:

若 s[i] 是小写字符，则 s[i+1] 不可以是相同的大写字符。

若 s[i] 是大写字符，则 s[i+1] 不可以是相同的小写字符。

请你将字符串整理好，每次你都可以从字符串中选出满足上述条件的 两个相邻 字符并删除，直到字符串整理好为止。

请返回整理好的 字符串 。题目保证在给出的约束条件下，测试样例对应的答案是唯一的。

注意：空字符串也属于整理好的字符串，尽管其中没有任何字符。

示例 1：

输入：s = "leEeetcode"

输出："leetcode"

解释：无论你第一次选的是 i = 1 还是 i = 2，都会使 "leEeetcode" 缩减为 "leetcode" 。

示例 2：

输入：s = "abBAcC"

输出：""

解释：存在多种不同情况，但所有的情况都会导致相同的结果。例如：

"abBAcC" --> "aAcC" --> "cC" --> ""

"abBAcC" --> "abBA" --> "aA" --> ""

示例 3：

输入：s = "s"

输出："s"

提示：

1 <= s.length <= 100

s 只包含小写和大写英文字母

# 分析

## 方法一：模拟/栈

思路：

可以使用栈来模拟整理字符串的过程。遍历字符串，对于每个字符，如果栈顶字符和当前字符满足删除条件，则将栈顶字符弹出，否则将当前字符压入栈中。最后将栈中字符拼接起来即可得到整理好的字符串。具体实现如下：

class Solution {

public:

string makeGood(string s) {

stack<char> stk;

for (char c : s) {

if (!stk.empty() && abs(stk.top() - c) == 32) {

stk.pop();

} else {

stk.push(c);

}

}

string result = "";

while (!stk.empty()) {

result = stk.top() + result;

stk.pop();

}

return result;

}

};

或：

class Solution {

public:

string makeGood(string s) {

string ret;

for (char ch: s) {

if (!ret.empty() && tolower(ret.back()) == tolower(ch) && ret.back() != ch) {

ret.pop\_back();

}

else {

ret.push\_back(ch);

}

}

return ret;

}

};

**复杂度分析：**

时间复杂度：O(N)，其中N为字符串 s 的长度。我们要遍历字符串 s 的每一个字符，而对每个字符都只需要常数时间的操作。

空间复杂度：O(N)或 O(1)，取决于使用语言的字符串类型是否是可修改的。