题目

A B C D E F G H I J

去做题

H 剪刀石头布

时间限制：1000ms 内存限制：65536kb

通过率：26/50 (52.00%) 正确率：26/237 (10.97%)

题目描述

妮妮发现了一个剪刀石头布的新玩法，初始给定一个长度为 n 的剪刀石头布序列 S，其中只有 $\mathbf{S}, \mathbf{R}, \mathbf{P}$ 三种字符，分别表示剪刀、石头与布。游戏会进行 $n-1$ 轮，每一轮会根据上一轮的序列生成一个新的剪刀石头布序列，每一轮操作按如下方式进行：

- 设上一轮的序列（若为第一轮则为初始给定的序列）为 S，这一轮生成的新序列为 T，有 $|T| = |S| - 1$。
- 对于 $i = 1, 2, \ldots, n-1$，如果 $S_i = S_{i+1}$ 则有 $T_i = S_i$，否则 T_i 为 S_i 和 S_{i+1} 进行普通剪刀石头布游戏获胜的一方所对应的字符。

妮妮发现最后一轮所生成的序列仅包含一个字符，她想知道具体是哪个字符。

输入格式

第一行一个正整数 t（$1 \le t \le 50$），表示数据组数。

对于每组数据，一行一个由 $\mathbf{S}, \mathbf{R}, \mathbf{P}$ 组成的字符串 S（$1 \le |S| \le 10^6$），含义同题目描述。

对于所有数据，保证 $|S|$ 之和不超过 3×10^6。

输出格式

对于每组数据，输出一行一个字符，表示最后一轮生成序列所包含的字符。

思路

用栈进行模拟

通过观察几个例子，如SSSSSSSSSPPPPPPPPRRRRRRRRPPPPPPPPSSSSSSSRRRRRRR能得到一些规律，可以用栈操作模拟这些规律 我的思路是用两个计数器，lose,win分别记录赢和输的次数；

赢和输如何判断?

遍历字符串，当前字符串和前一个比较，前一个赢为赢，前一个输为输；

赢的情况

win++;把当前字符入栈

输的情况

lose++;栈顶即前一个字符出栈 然后比较lose和win的数量，若lose>win，需要清空栈，同时清零计数器win = 0;lose = 0;再将当前字符入栈

代码实现

```
#include <iostream>
#include <stack>
using namespace std;

bool winner(char a, char b) {
    if ((a == 'S' && b == 'P') || (a == 'P' && b == 'R') || (a == 'R' && b == 
'S')) {
        return true;
    } else {
        return false;
    }
}

char lastCharacter(const string& s) {
    int len = s.size() - 1;
    stack<char> st;
    st.push(s[0]);
    int i = 1;
    int win = 0;
    int lose = 0;

    while(i <= len && winner(st.top(),s[i])){
            st.push(s[i]);
            win ++;
            i ++;
    }
    if(i == len + 1) {
        char c;
        while(!st.empty()){
            c = st.top();
            st.pop();
        }
        return c;
```

```
    }

    for (int j = i;j <=len ; j ++) {
        if(!st.empty()&&winner(st.top(),s[j]) == false){
            lose ++;
            st.pop();
            if(lose > win){
                while(!st.empty()){
                    st.pop();
                };
                st.push(s[j]);
                win = 0;
                lose = 0;
            }
            else
                continue;
        }else{
            win ++;
            st.push(s[j]);
        }
    }
    char c;
    while(!st.empty()){
        c = st.top();
        st.pop();
    }
    return c;
}

int main() {
    int t;
    cin >> t;

    while (t--) {
        string s;
        cin >> s;
        string newS = "";
        newS += s[0];
        int k = 0;
        for(int i = 0;i < s.size(); i ++){
            if(s[i] != newS[k]){
                newS += s[i];
                k ++;
            }
        }
        s = newS;
        cout << lastCharacter(s) << endl;
    }

    return 0;
}
```

要注意的细节

本题可以先将字符串进行去重复处理，优化时间