比较含退格的字符串

给定 S 和 T 两个字符串，当它们分别被输入到空白的文本编辑器后，判断二者是否相等，并返回结果。 # 代表退格字符。

注意：如果对空文本输入退格字符，文本继续为空。

示例 1：

输入：S = "ab#c", T = "ad#c"

输出：true

解释：S 和 T 都会变成 “ac”。

示例 2：

输入：S = "ab##", T = "c#d#"

输出：true

解释：S 和 T 都会变成 “”。

示例 3：

输入：S = "a##c", T = "#a#c"

输出：true

解释：S 和 T 都会变成 “c”。

示例 4：

输入：S = "a#c", T = "b"

输出：false

解释：S 会变成 “c”，但 T 仍然是 “b”。

C 语言解法：双指针　退格字符长度比较　以及原地修改字符串

```
/*采用双指针，遇到‘#’，下标减一，删除前一个字符；其他字符依次增加下标计数；得到新的字符串，然后就是字符串是否相同对比*/
int deletebackspace(char *str,int size)
{
    if(str==NULL||size==0)
    {
        return 0;
```

```
    }
    int index = 0;
    for(int i=0;i<size;i++)
    {
        if(str[i] != '#')
        {
            str[index++] = str[i];
        }
        else
        {
            // str[i] == '#'
            if(index>0)
            {
                index--;//遇到# 回退一个索引  下一次赋值直接覆盖  实现#前字符删除  注意 字符串末尾有一个'\0'
            }
        }
    }
    return index;
}

bool backspaceCompare(char * S, char * T){
    int slen = deletebackspace(S,strlen(S));
    int tlen = deletebackspace(T,strlen(T));
    if(slen != tlen)
    {
        return false;
    }
    else
    {
        for(int i=0;i<slen;i++)
        {
            if(S[i] != T[i])
            {
                return false;
            }
        }
        return true;
    }
}
```

C++：栈实现

```
class Solution {
public:
    string buildnewstr(string str,int n)
    {
        string st;//string 类模拟一个栈实现
        for(int i=0;i<n;i++)
        {
            if(str[i]!='#')
                st.push_back(str[i]);
            else if(!st.empty())
            {
                st.pop_back();//非空弹出栈顶
            }
        }
        return st;
    }

    bool backspaceCompare(string S, string T) {
        //通过栈实现
        //遇到‘#’，如果栈不为空  弹出栈顶
        //遇到非‘#’,直接入栈
        int len1=S.size();
        int len2=T.size();
        return buildnewstr(S,len1)==buildnewstr(T,len2);
    }
};
```