CHAR ARRAY 1

```
char arr[10];
cout << "Enter your Name" << endl;
cin >> arr; → Kash
cout << "Your name is : " << arr << endl;
```

Loops se bhi input le sakte hai but no need we have 'cin'

arr[0] → K
arr[1] → a
arr[2] → s
arr[3] → h
arr[4] → /0

Default delimiters

↳ Enter } '\n'
↳ tab } '\t'
↳ space } "_"

inme se kuch bhi hit hua toh input lena band ho jayega

⇒ we can use getline() function → isme koi delimiter ka kuch asar nahi padega

⇒ cin.getline(arr, 100);

↳ cin.getline('arr', 'no.of.characters', delimiter)

★★ agar yeh aaya (detect hua) toh getline terminate ho jayega

CHAR ARRAY AND STRING 2

LEETCODE 1047

#REMOVE ALL ADJACENT DUPLICATE IN A STRING

① i/p → string → "abbaca"

o/p → "ca"

↳ ① → abbaca

② → aaca

③ → ca

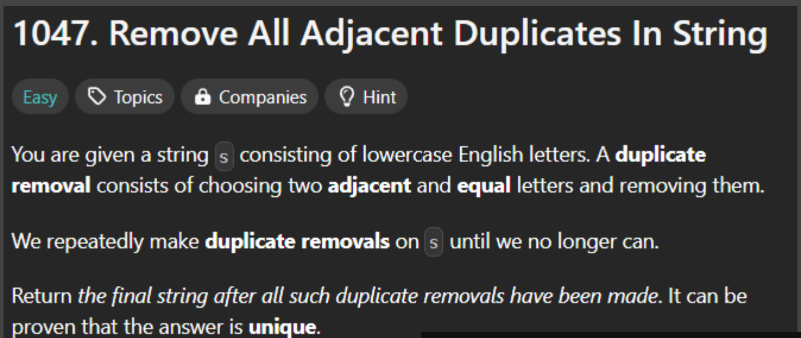
② i/p → string → "azxxzy"

o/p → ""

↳ ① → azxxzy

② → azzy

③ → ay

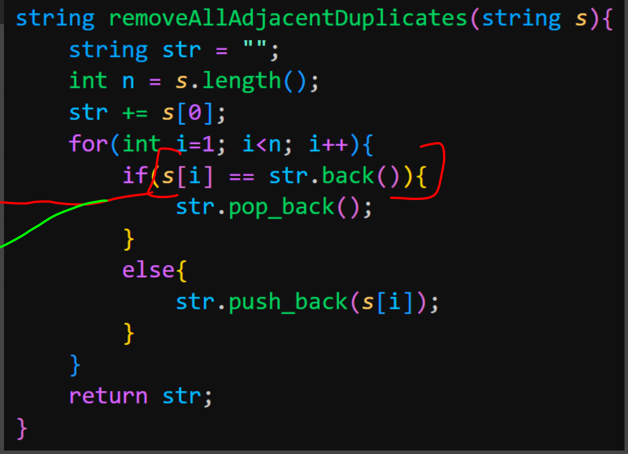


1047. Remove All Adjacent Duplicates In String

You are given a string `s` consisting of lowercase English letters. A **duplicate removal** consists of choosing two **adjacent** and **equal** letters and removing them.

We repeatedly make **duplicate removals** on `s` until we no longer can.

Return *the final string after all such duplicate removals have been made*. It can be proven that the answer is **unique**.

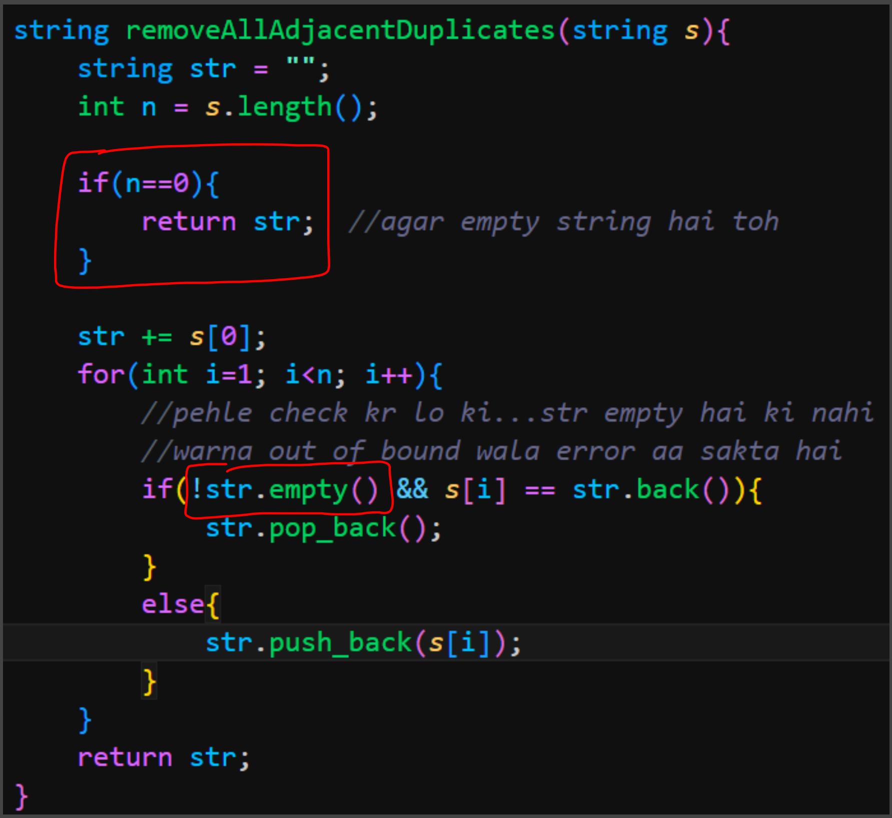


```
string removeAllAdjacentDuplicates(string s){
    string str = "";
    int n = s.length();
    str += s[0];
    for(int i=1; i<n; i++){
        if(s[i] == str.back()){
            str.pop_back();
        }
        else{
            str.push_back(s[i]);
        }
    }
    return str;
}
```

[out of bound aa sakta ha]

↳ empty check kro

[!str.empty]



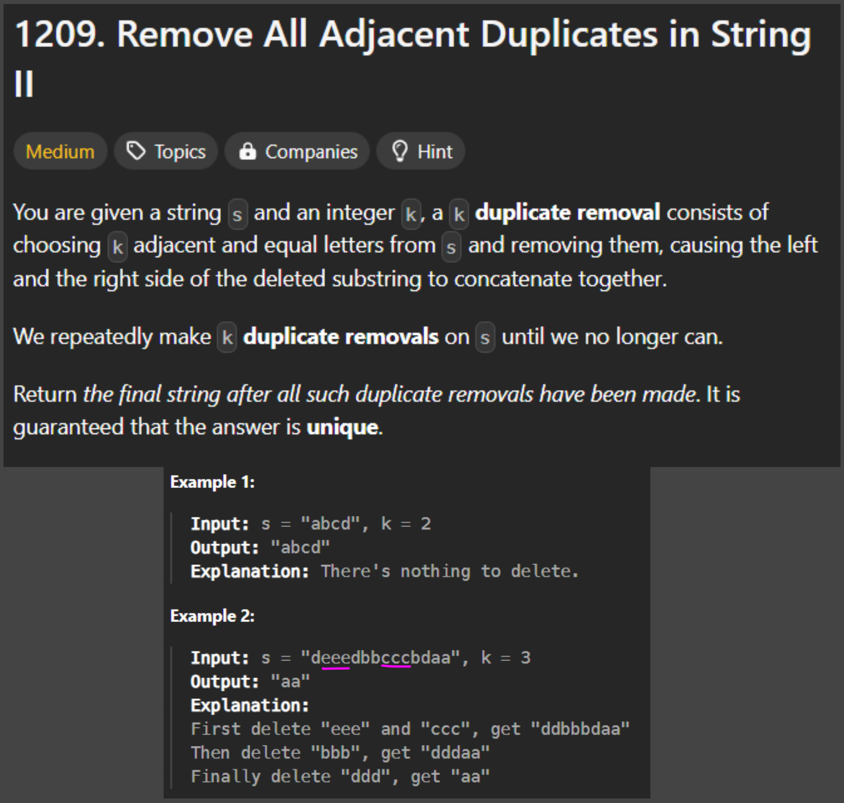
```cpp
string removeAllAdjacentDuplicates(string s){
    string str = "";
    int n = s.length();

    if(n==0){
        return str;  //agar empty string hai toh
    }

    str += s[0];
    for(int i=1; i<n; i++){
        //pehle check kr lo ki...str empty hai ki nahi
        //warna out of bound wala error aa sakta hai
        if(!str.empty() && s[i] == str.back()){
            str.pop_back();
        }
        else{
            str.push_back(s[i]);
        }
    }
    return str;
}
```

LEETCODE 1209

REMOVE ALL ADJACENT DUPLICATE IN STRING 2



1209. Remove All Adjacent Duplicates in String II

Medium | Topics | Companies | Hint

You are given a string `s` and an integer `k`, a `k` **duplicate removal** consists of choosing `k` adjacent and equal letters from `s` and removing them, causing the left and the right side of the deleted substring to concatenate together.

We repeatedly make `k` **duplicate removals** on `s` until we no longer can.

Return *the final string after all such duplicate removals have been made*. It is guaranteed that the answer is **unique**.

Example 1:

```
Input: s = "abcd", k = 2
Output: "abcd"
Explanation: There's nothing to delete.
```

Example 2:

```
Input: s = "deeedbbcccbdaa", k = 3
Output: "aa"
Explanation:
First delete "eee" and "ccc", get "ddbbbdaa"
Then delete "bbb", get "dddaa"
Finally delete "ddd", get "aa"
```

deeedbbccc bdaa

ddbbbdaa

dddaa

aa ✓