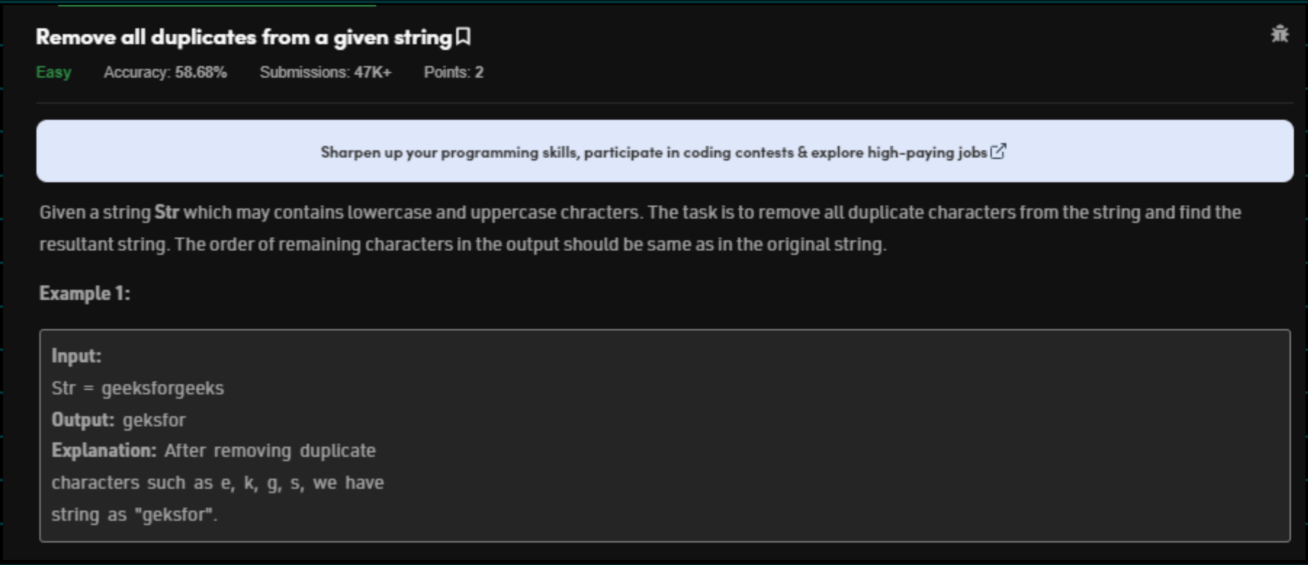
Print or remove duplicate from given string



Remove all duplicates from a given string

Easy Accuracy: 58.68% Submissions: 47K+ Points: 2

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given a string **Str** which may contains lowercase and uppercase chracters. The task is to remove all duplicate characters from the string and find the resultant string. The order of remaining characters in the output should be same as in the original string.

Example 1:

```
Input:
Str = geeksforgeeks
Output: geksfor
Explanation: After removing duplicate
characters such as e, k, g, s, we have
string as "geksfor".
```

In this problem we have to remove the duplicate from string and keeping the order same.

So first thing that come in mind is Hashmap/unordered_map

we we store all the element in unordered_map with the counting frequency of element

Str = "geeksforgeeks"

'g'	:	2
'e'	:	4
'k'	:	2
's'	:	1
'f'	:	1
'o'	:	1
'r'	:	1

After this run another loop

loop(i = 0 -> str.size)

Check if the freq is ≥ 1

then insert that value in our answer and make freq in map=0 because we have already visited that element

geeks for geeks

check (mp[str[i]]≥1)

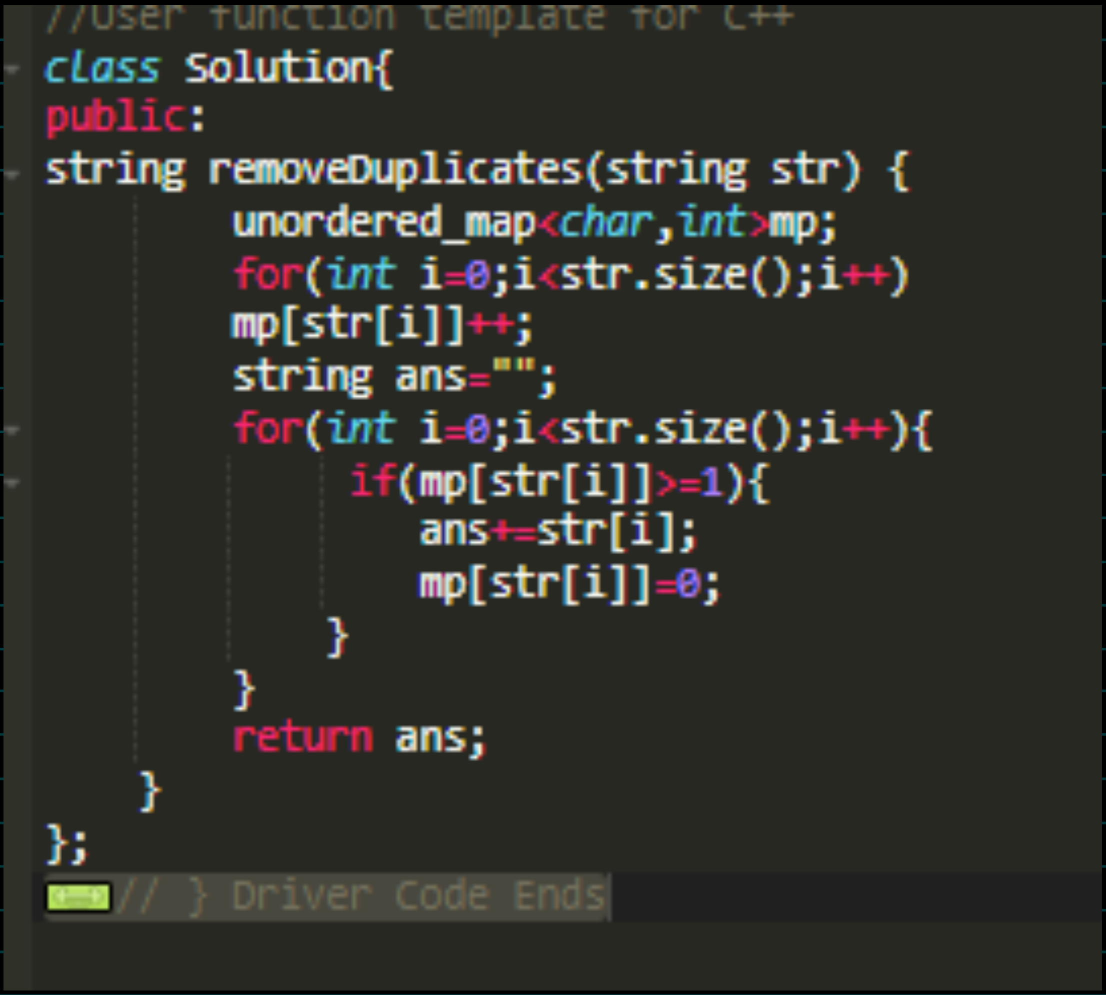
True

ans += str[i]

make mp[str[i]]=0

'g'	:	~~2~~ 0
'e'	:	~~4~~ 0
'k'	:	~~2~~ 0
's'	:	~~1~~ 0
'f'	:	~~1~~ 0
'o'	:	~~1~~ 0
'r'	:	~~1~~ 0

ans = "geksfor"



```
//User function template for C++
class Solution{
public:
string removeDuplicates(string str) {
        unordered_map<char,int>mp;
        for(int i=0;i<str.size();i++)
        mp[str[i]]++;
        string ans="";
        for(int i=0;i<str.size();i++){
            if(mp[str[i]]>=1){
                ans+=str[i];
                mp[str[i]]=0;
            }
        }
        return ans;
    }
};
// } Driver Code Ends
```

To print All the duplicate

Change condition to   geeksfor

if (mp[str[i]] ≥ 2)

cout << str[i]; // gek

'g'	:	2
'e'	:	4
'k'	:	2
's'	:	1
'f'	:	1
'o'	:	1
'r'	:	1

T.C – O(n+n)

S.C – O(n)

NOTES: This problem have another variation lexicographical order.