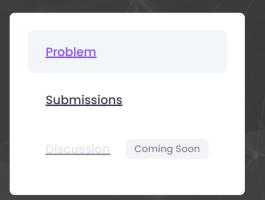


## **Balanced Brackets**

• Contest List • Algorithm Competition Summer Camp 2023 Foundation Upsolving Contest • Problem List • Balanced Brackets • Problem



A bracket is considered to be any one of the following characters: (, ), {, }, [, or ].

A balanced bracket sequence is a string consisting of only brackets, such that every opening bracket has its closing bracket.

Formally you can define a balanced bracket sequence with: the empty string is a balanced bracket sequence. if "s" is a balanced bracket sequence, then so is (s), {s} and [s]. if "s" and t are balanced bracket sequences, then so is st.

Given a string of brackets, determine whether the string of brackets is a balanced bracket sequence. If a string is a balanced bracket sequence, print **YES**. Otherwise, print **NO**.

Input Format	
A single-line string.	
Output Format	
If a string is a balanced bracket sequence, print <b>YES</b> . Otherwise, print <b>NO</b> .	
Constraints	
$1 \leq$ length of the string $\leq 2*10^5$	
Sample Input 1 🔲	Sample Output 1 🔲
{[()]}	YES
Sample Input 2 🔲	Sample Output 2 🔲
{[(])}	NO
Sample Input 3 🖵	Sample Output 3 🔲
{{[[((())]]}}	YES

```
C++ (GCC 9.2.0)
                                       Bright 💙
                                                                             Memory Limit (kB): 256000 Time Limit (s):1
    #include <iostream>
     #include <stack>
     #include <string>
    using namespace std;
 6
 7 - bool arePair(char opening, char closing) {
          if(opening == '(' && closing == ')') return true;
else if(opening == '{' && closing == '}') return true;
else if(opening == '[' && closing == ']') return true;
 8
 9
10
           return false;
11
12 }
13
11 - hool anaRnackateRalancad(etd..etning avnn) 5
```

```
וא דא טטטן מו בחו מרעבר באמורבת ( בירו זווא באלו ) ל
15
          stack<char> S;
16
          for(int i =0; i < expr.length(); i++) {
   if(expr[i] == '(' || expr[i] == '{' || expr[i] == '[')}</pre>
17 -
18
                   S.push(expr[i]);
19
20
              else if(expr[i] == ')' || expr[i] == '}' || expr[i] == ']') {
   if(S.empty() || !arePair(S.top(), expr[i]))
21 🔻
22
23
                        return false;
24
                   else
25
                        S.pop();
26
27
28
          return S.empty() ? true:false;
29
30 }
31
                                  Test against custom test case
   1 Upload File
                                                                                     Run Code
                                                                                                      Submit
                                     Accepted
   ✓ <u>Sample Test Case 0</u>
                                    Input(stdin)
   ✓ <u>Sample Test Case 1</u>
                                        1 {{[[(())]]}}
      Sample Test Case 2
                                     Output(stdin)
                                       1 YES
                                        2
                                     Expected Output
                                     1 YES
```

algoleague © All Rights Reserved 2023 - v1.3.3

About Help Legal Contact Us