Balanced Brackets

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 **YES**. Otherwise, print **NO**.

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;

bool arePair(char opening, char closing) {
    if(opening == '(' && closing == ')') return true;
    else if(opening == '{' && closing == '}') return true;
    else if(opening == '[' && closing == ']') return true;
    return false;
}

bool areBracketsBalanced(std::string expr) {
```

```cpp
bool areBracketsBalanced(std::string expr) {
    stack<char> S;

    for(int i =0; i < expr.length(); i++) {
        if(expr[i] == '(' || expr[i] == '{' || expr[i] == '[')
            S.push(expr[i]);

        else if(expr[i] == ')' || expr[i] == '}' || expr[i] == ']') {
            if(S.empty() || !arePair(S.top(), expr[i]))
                return false;
            else
                S.pop();
        }
    }

    return S.empty() ? true:false;
}

```

Upload File

Test against custom test case

Run Code

Submit

✓ Sample Test Case 0

✓ Sample Test Case 1

✓ Sample Test Case 2

Accepted

Input(stdin)

```
{{[[(())]]}}

```

Output(stdin)

```
YES

```

Expected Output

```
YES
```