Fidelity International Python Developer Hiring Test - Exp... 14m to test end 0/4 Attempted Vivek Ratnaparkhi

1

2

3

4

☆ Braces

You are designing a compiler for a C++ program and need to check that braces in any given file are balanced.

Braces in a string are considered to be balanced if the following criteria are met:

- All braces must be closed. Braces come in pairs of the form (), {} and []. The left brace *opens* the pair, and the right one *closes* it.
- In any set of nested braces, the braces between any pair must be closed.

For example, [{}] is a valid grouping of braces but [}]{} is not.

Function Description

Complete the function *braces* in the editor below. The function must return an array of strings where the string at each index *i* denotes whether or not the braces were balanced in a $values_i$. The array should consist of strings "YES" or "NO" aligned with their indexes in *values*.

braces has the following parameter(s):

$values[values_0, \ldots values_{n-1}]$: an array of strings to analyze

Constraints

- $1 \le n \le 15$
- $1 \le$ length of $values_i \le 100$
- It is guaranteed that each $values_i$ consists of (,), {, }, [, and] only.

Input Format For Custom Testing

Sample Case 0

Sample Input For Custom Testing

```
2
{}[]()
{[}]}
```

Sample Output

```
YES
NO
```

Explanation

$values_0$: {}[]() meets the criteria for a balanced string, so index *0* in our return array should contain the string YES.

$values_1$: {[}]} contains unmatched braces between a matched pair in the substrings [}, {[}, and [}], so index *1* in our return array should contain the string NO.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ✕

For help on how to read input and write output in Python 3, click here. ×

Original Code Python 3

Fidelity International Python Developer Hiring Test - Exp...

14m
to test end

0/4 Attempted

Vivek Ratnaparkhi

1

2

3

4

```
    opening_braces = ['{', '[', '(']
    closing_braces = ['}', ']', ')']

    stack = []
    result = []

    for braces in values:
        for b in braces:
            if not stack:
                stack.append(b)
            else:
                if opening_braces.index(stack[-1]) == closing_braces.index(b):
                    stack.pop()
                else:
                    stack.append(b)

        if stack:
            result.append('NO')
        else:
            result.append('YES')

    return result

```

Line: 10 Col: 1

□ Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

About Privacy Policy Terms of Service