题目描述

给定一个字符串，里边可能包含"()"、"[]"、"{}"三种括号，请编写程序检查该字符串中的括号是否成对出现，且嵌套关系正确。
若括号成对出现且嵌套关系正确，或该字符串中无括号字符，输出：true；
若未正确使用括号字符，输出：false。
实现时，无需考虑非法输入。

输入描述

无

输出描述

无

用例

输入	(1+2)/(0.5+1)
输出	true
说明	无

题目解析

JavaScript算法源码

```javascript
// JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(line) {
  const regExp = /[^\(\)\[\]\{\}]/g;

  // 去除非括号字符
  line = line.replace(regExp, "");

  const map = {
    ")": "(",
    "]": "[",
    "}": "{",
  };

  const stack = [];

  for (let c of line) {
    if (stack.length && map[c]) {
      if (stack.at(-1) === map[c]) {
        stack.pop();
        continue;
      } else {
        return false;
      }
    }

    stack.push(c);
  }

  return stack.length == 0;
}
```

Java算法源码

```java
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
  // 输入获取
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String s = sc.nextLine();

    System.out.println(getResult(s));
  }

  // 算法入口
  public static boolean getResult(String s) {
    // 去除非括号字符
    s = s.replaceAll("[^\\(\\)\\[\\]\\{\\}]", "");

    HashMap<Character, Character> map = new HashMap<>();
    map.put(')', '(');
    map.put(']', '[');
    map.put('}', '{');

    LinkedList<Character> stack = new LinkedList<>();

    for (int i = 0; i < s.length(); i++) {
      char c = s.charAt(i);

      if (stack.size() > 0 && map.containsKey(c)) {
        if (stack.getLast() == map.get(c)) {
          stack.removeLast();
          continue;
        } else {
          return false;
        }
      }

      stack.add(c);
    }

    return stack.size() == 0;
  }
}
```

Python算法源码

```python
import re

# 输入获取
s = input()


# 算法入口
def getResult(s):
    # 去除非括号字符
    s = re.sub(r"[^\(\)\[\]\{\}]", "", s)

    map = {
        ")": "(",
        "]": "[",
        "}": "{"
    }

    stack = []

    for i in range(len(s)):
        c = s[i]

        if len(stack) > 0 and map.get(c) is not None:
            if stack[-1] == map[c]:
                stack.pop()
                continue
            else:
                return False

        stack.append(c)

    return len(stack) == 0


# 算法调用
print(str(getResult(s)).lower())
```