# Balanced Brackets

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

Two brackets are considered to be a *matched pair* if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e., ), ], or }) *of the exact same type*. There are three types of matched pairs of brackets: [], {}, and ().

A matching pair of brackets is *not balanced* if the set of brackets it encloses are not matched. For example, {[(])} is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket, ].

By this logic, we say a sequence of brackets is *balanced* if the following conditions are met:

* It contains no unmatched brackets.
* The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets.

Given n strings of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, return YES. Otherwise, return NO.

**Function Description**

Complete the function *isBalanced* in the editor below. It must return a string: YES if the sequence is balanced or NO if it is not.

isBalanced has the following parameter(s):

* *s*: a string of brackets

**Input Format**

The first line contains a single integer n, the number of strings.   
Each of the next n lines contains a single string s, a sequence of brackets.

**Constraints**

* 1<= n<= 10^3
* 1<= |s| <=10^3, where s is the length of the sequence.
* All chracters in the sequences ∈ { **{**, **}**, **(**, **)**, **[**, **]** }.

**Output Format**

For each string, return YES or NO.

**Sample Input**

3

{[()]}

{[(])}

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

**Sample Output**

YES

NO

YES

**Explanation**

1. The string {[()]} meets both criteria for being a balanced string, so we print YES on a new line.
2. The string {[(])} is not balanced because the brackets enclosed by the matched pair { and } are not balanced: [(]).
3. The string {{[[(())]]}} meets both criteria for being a balanced string, so we print YES on a new line.

using System.CodeDom.Compiler;

using System.Collections.Generic;

using System.Collections;

using System.ComponentModel;

using System.Diagnostics.CodeAnalysis;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Runtime.Serialization;

using System.Text.RegularExpressions;

using System.Text;

using System;

class Solution {

// Complete the isBalanced function below.

static string isBalanced(string s) {

int stringlength =s.Length;

if (stringlength%2!=0) return "NO";

Stack q1=new Stack();

Stack q2=new Stack();

Stack q3=new Stack();

int number;

for(int i=0; i<stringlength;i++){

if(s[i]=='{')

{

q1.Push(i);

}

if(s[i]=='}'){

if(q1.Count==0)return "NO";

number=(int)q1.Peek();

if(q2.Count>0){

if((int)q2.Peek()<i && (int)q2.Peek()>number)return "NO";

}

if(q3.Count>0){

if((int)q3.Peek()<i && (int)q3.Peek()>number)return "NO";

}

q1.Pop();

}

if(s[i]=='(')

{

q2.Push(i);

}

if(s[i]==')'){

if(q2.Count==0)return "NO";

number=(int)q2.Peek();

if(q1.Count>0){

if((int)q1.Peek()<i && (int)q1.Peek()>number)return "NO";

}

if(q3.Count>0){

if((int)q3.Peek()<i && (int)q3.Peek()>number)return "NO";

}

q2.Pop();

}

if(s[i]=='[')

{

q3.Push(i);

}

if(s[i]==']'){

if(q3.Count==0)return "NO";

number=(int)q3.Peek();

if(q2.Count>0){

if((int)q2.Peek()<i && (int)q2.Peek()>number)return "NO";

}

if(q1.Count>0){

if((int)q1.Peek()<i && (int)q1.Peek()>number)return "NO";

}

q3.Pop();

}

}

if(q1.Count>0||q1.Count>0||q3.Count>0) return "NO";

return"YES";

}

static void Main(string[] args) {

TextWriter textWriter = new StreamWriter(@System.Environment.GetEnvironmentVariable("OUTPUT\_PATH"), true);

int t = Convert.ToInt32(Console.ReadLine());

for (int tItr = 0; tItr < t; tItr++) {

string s = Console.ReadLine();

string result = isBalanced(s);

textWriter.WriteLine(result);

}

textWriter.Flush();

textWriter.Close();

}

}

Submit CodeRun Code

Upload Code as File



**Test against custom input**

Problem Solving

You have earned 25.00 points!

These points will also count towards your progress in the Problem Solving Badge.

**100%**

**Congratulations**

You solved this challenge. Would you like to challenge your friends?

[Next Challenge](https://www.hackerrank.com/challenges/ctci-queue-using-two-stacks?h_l=interview&playlist_slugs%5B%5D=interview-preparation-kit&playlist_slugs%5B%5D=stacks-queues&h_r=next-challenge&h_v=zen)

* **Test case 0**
* **Test case 1**
* **Test case 2**
* **Test case 3**
* **Test case 4**
* **Test case 5**
* **Test case 6**
* **Test case 7**
* **Test case 8**
* **Test case 9**
* **Test case 10**
* **Test case 11**
* **Test case 12**
* **Test case 13**
* **Test case 14**
* **Test case 15**
* **Test case 16**
* **Test case 17**
* **Test case 18**
* **Test case 19**
* **Test case 20**