Brackets Balanced

Send Feedback

For a given a string expression containing only round brackets or parentheses, check if they are balanced or not. Brackets are said to be balanced if the bracket which opens last, closes first.

Example:

Expression: (()())
Since all the opening brackets have their corresponding closing brackets, we say it is balanced and hence the output will be, 'true'.

You need to return a boolean value indicating whether the expression is balanced or not.

Note:

The input expression will not contain spaces in between.

Input Format:

The first and the only line of input contains a string expression without any spaces in between.

Output Format:

The only line of output prints 'true' or 'false'.

Note:

You don't have to print anything explicitly. It has been taken care of. Just implement the function.

Constraints:

1 <= N <= 10^7
Where N is the length of the expression.

Time Limit: 1sec

Sample Input 1 :

(()()())

Sample Output 1 :

true

Sample Input 2 :

()()(()

Sample Output 2 :

false

Explanation to Sample Input 2:

The initial two pairs of brackets are balanced. But when you see, the opening bracket at the fourth index doesn't have its corresponding closing bracket which makes it imbalanced and in turn, making the whole expression imbalanced. Hence the output prints 'false'.

```
import java.util.*;
public class Solution {

   public static boolean isBalanced(String expression) {
      //Your code goes here//

      Stack<Character> s=new Stack<>();



    for(int i=0;i<expression.length();i++)
    {
       if(expression.charAt(i)=='(')
       {
          s.push('(');
       }
```

```
        else
        {
           if(s.isEmpty()==true)
           {
              return false;
           }
           else
           {
              if(s.peek()=='(' && expression.charAt(i)==')')
              {
                 s.pop();
              }
           }
        }
     }
      if(s.isEmpty())
      {
         return true;
      }
      return false;

   }
}
```