1. Java: Braces

Given a list of strings of bracket characters: *{}()*, the string of brackets is *balanced* under the following conditions:

1. It is the empty string.
2. If strings *a* and *b* are balanced, then *ab* is balanced.
3. If string *a* is balanced, then *(a)* and *{a}* are balanced.

Write a class that determines whether the brackets in each string are balanced and returns *true* if the string is balanced, or *false* if it is not.

Example 0

s = [*"{}()", "{()}", "({()})"*]

s[0] exhibits condition *2* above. "{}" and "()" are balanced, so "{}()" is balanced. Return *true*.
s[1] exhibits condition 3 above. "()" is balanced, so "{()}" is balanced. Return *true*.
s[2] exhibits condition 3 above. "()" is balanced, so "{()}" is balanced and "({()})" is balanced. Return *true*.

Example 1

s = [*"{}(", "(()}", "((", "}{"*]

Example 1

s = ["{}(", "({})", "((", "}{"]

s[0] → "{}(" is an unbalanced string due to the open "(". Return *false*.

s[1] → "({})" is an unbalanced string due to ")" before "{" has been closed. Return *false*.

s[2] → "((", is an unbalanced string because neither "(" is closed. Return *false*.

s[2] → "}{" is an unbalanced string because "}" comes before a "{" and because the final "{" is not closed. Return *false*.

Function Description

The provided code contains the declaration for a class named *Solution* with a *main* method that does the following:

- Creates a *Parser* object.
- Reads an unknown number of strings from stdin.
- Passes each string as an argument to the *Parser* object's *isBalanced* method and prints value returned by the method on a new line.

Complete the function an *isBalanced* in the editor below.

isBalanced has the following parameter(s):

2. Java: Shape

Define the following 2 classes to represent 2-dimensional objects.

Super Class: *Shape*
It should have

- 2 member variables: *length*, and *breadth* of integer types.
- 2 argument constructor for *length* and *breadth* which stores the arguments in their corresponding member variables.
- A method, *area()*, which prints the *length* and *breadth* of the shape, delimited by a space.

Concrete Class: *Rectangle*
It should have

- 2 argument constructor for *length* and *breadth*. It should forward those arguments to the superclass constructor.
- Override the *area()* method to print the area using the formula *(length*breadth)*.

Concrete Class: *Rectangle*

It should have

- 2 argument constructor for *length* and *breadth*. It should forward those arguments to the superclass constructor.
- Override the *area()* method to print the area using the formula *(length*breadth)*.

Constraints

- $0 < length \leq 1000$
- $0 < breadth \leq 1000$

▼ Input Format For Custom Testing

A single line of input consists of two space-separated integers, *length*, and *breadth*.

▼ Sample Case 0

Sample Input For Custom Testing

```
4 5
```

Sample Output

```
4 5
20
```