```
str.replaceFirst("^0+(?!$)", "");
```

0002 → 0

→ ".r

remove leading Zeroes.

" ^0+(?!$) "

^ (caret) : match to the beginning of the string

0 : search zeroes.

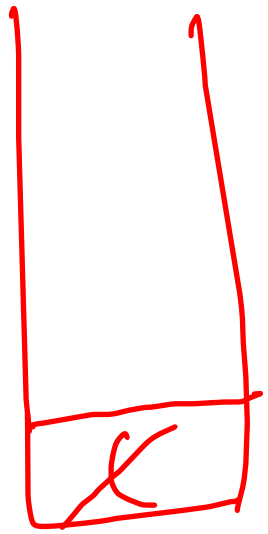
+ : count consecutive zeroes

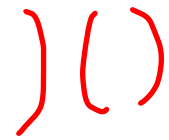
(?!$) : more char after zero, will print 0 even if all the digits are zero. Consider last digit.

(negative lookahead)

HW_Minimum Add to Make Parentheses Valid 3

close = 0 1





return close + size()

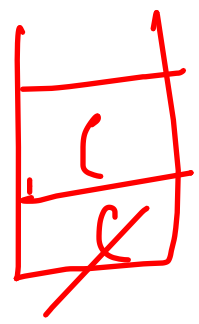
closing opening

① ()))(

② (())

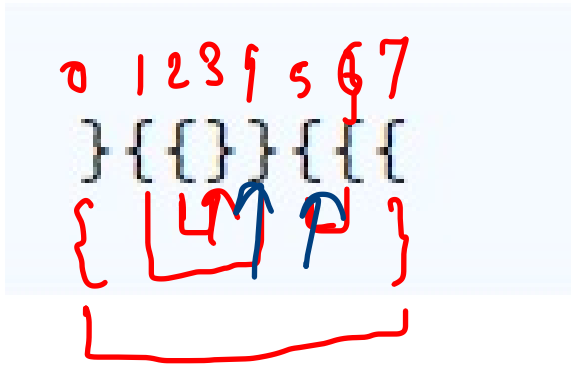
③))(

④ (()()(




```java
public static int valid(String s){

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

    int close = 0;

    char ch[] = s.toCharArray();

    for(char c : ch){
        if(c == '('){  // opening
            st.push(c);
        }
        else{  // closing
            if(!st.isEmpty() && st.peek() == '('){
                st.pop();
            }
            else{
                close++;
            }
        }
    }
    return st.size() + close;
}
```

O(n)

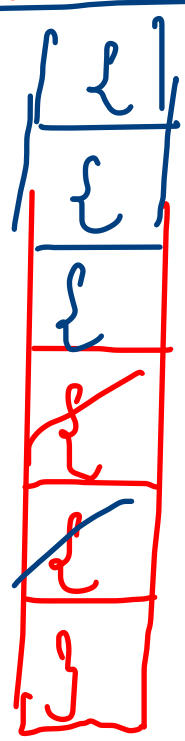


O(n)

2+1 3/2=1

2/2=1

1+1+1=3



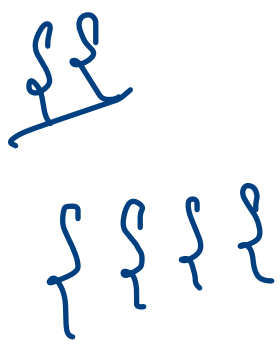
OC=0+2*3

CC=0+1

(3+1)/2 = 2 + 1 → 3

(1+1)/2 = 1

o/p



3

1

```
public static int reversal(String s){
    int len = s.length();

    if(len % 2 != 0){
        return -1;
    }

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

    for(char c : s.toCharArray()){
        if(c == '{'){
            st.push(c);
        }
        else{
            if(!st.isEmpty() && st.peek() == '{'){
                st.pop();
            }
            else{
                st.push(c);
            }
        }
    }
    int openCount = 0, closeCount = 0;
    while(!st.isEmpty()){
        if(st.pop() == '{'){
            openCount++;
        }
        else{
            closeCount++;
        }
    }

    return (openCount + 1)/2 + (closeCount + 1)/2;
}
```