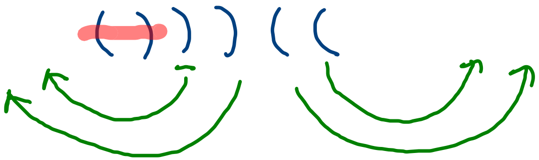
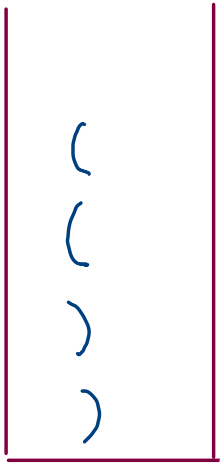
921. Minimum Add to Make Parentheses Valid



ans = 4



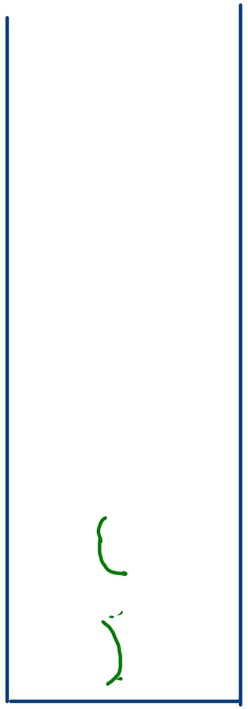
( ))) ((

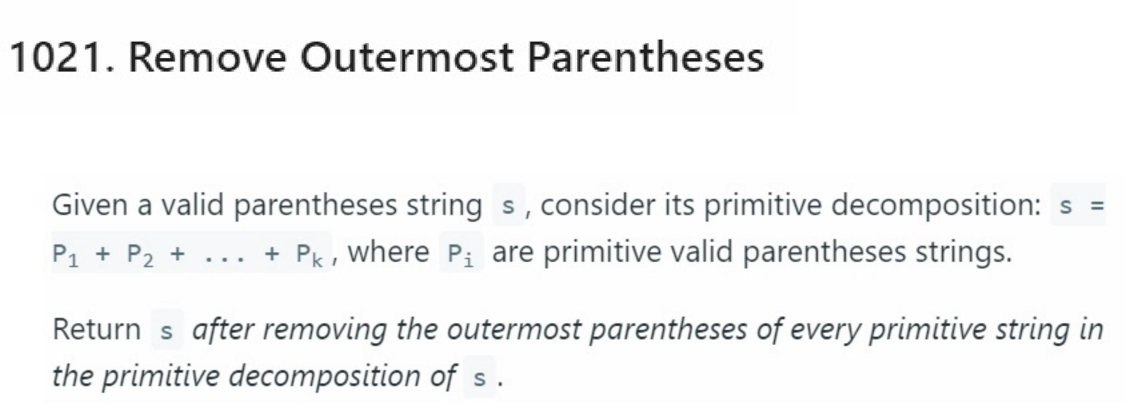
i

~~(()())~~)(~~()()~~

no ')' bracket for me

no '(' bracket for me

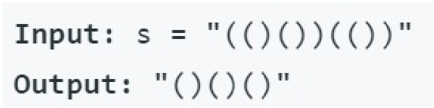




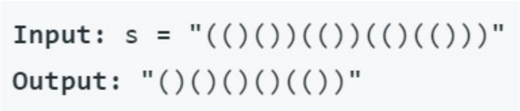
1021. Remove Outermost Parentheses

Given a valid parentheses string `s`, consider its primitive decomposition: `s = P1 + P2 + ... + Pk`, where `Pi` are primitive valid parentheses strings.

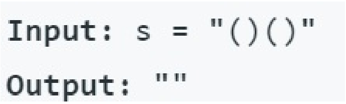
Return `s` *after removing the outermost parentheses of every primitive string in the primitive decomposition of* `s`.



```
Input: s = "(()())(())"
Output: "()()()"
```



```
Input: s = "(()())(())(()(()))"
Output: "()()()()(())"
```



```
Input: s = "()()"
Output: ""
```

$s = (()())(())$

$s = \cancel{(}()()\cancel{)} + \cancel{(}()\cancel{)}$

$ans = ()()()$

$s = (()())(())(()(()))$

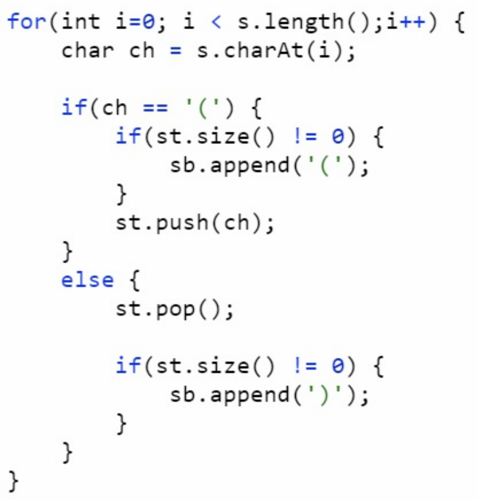
$s = \cancel{(}()()\cancel{)} + \cancel{(}()\cancel{)} + \cancel{(}()(())\cancel{)}$

$ans = ()()()()(())$

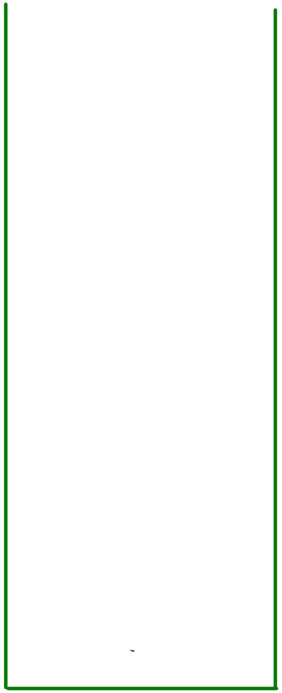
$s = ()()$

$s = \cancel{(}\cancel{)} + \cancel{(}\cancel{)}$

$ans =$ ""



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

    if(ch == '(') {
        if(st.size() != 0) {
            sb.append('(');
        }
        st.push(ch);
    }
    else {
        st.pop();

        if(st.size() != 0) {
            sb.append(')');
        }
    }
}
```



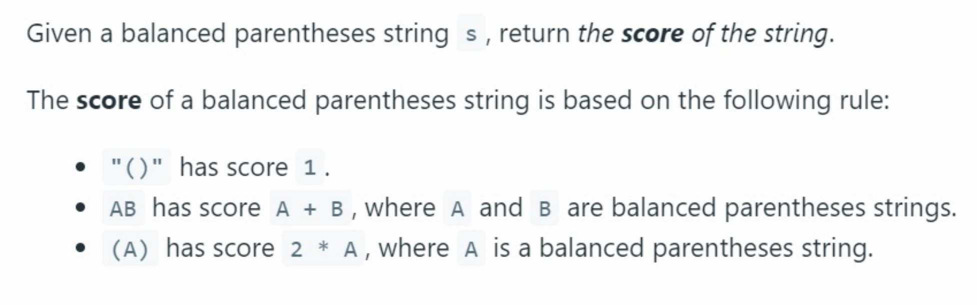
s = (( )) (( ) (( )))

i

s = ~~(~~ ( ) ~~)~~ + ~~(~~ ( ) (( )) ~~)~~

sb = ( ) ( ) (( ))

856. Score of Parentheses



Given a balanced parentheses string `s`, return *the **score** of the string*.

The **score** of a balanced parentheses string is based on the following rule:

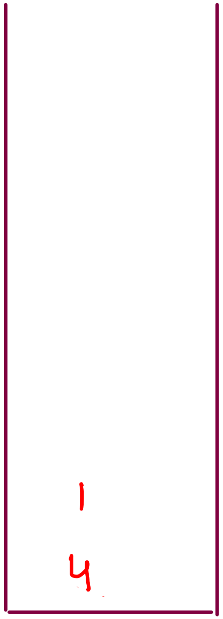
- `"()"` has score `1`.
- `AB` has score `A + B`, where `A` and `B` are balanced parentheses strings.
- `(A)` has score `2 * A`, where `A` is a balanced parentheses string.

( ) → 1

A B → score of A + score of B

(A) → 2 * score of A

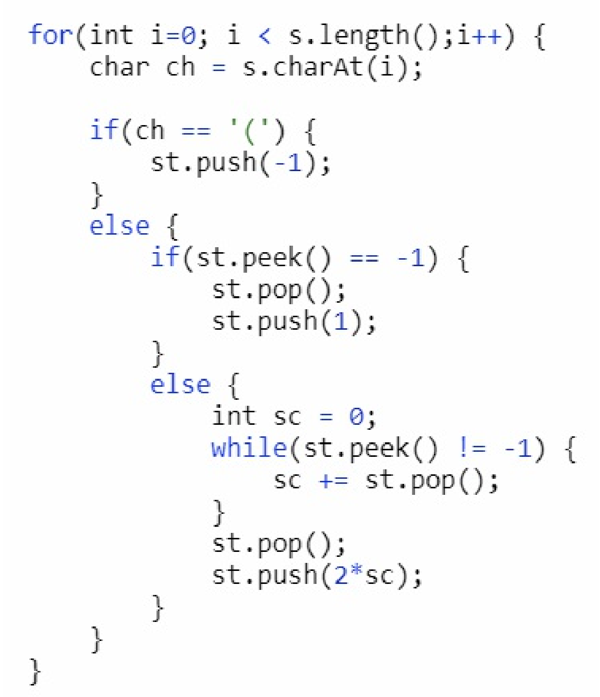
$$\underbrace{(()())}_{4}\ \underbrace{(\ )}_{1}\ i$$



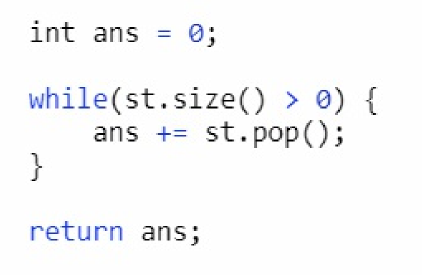
ans = 1 + 4

st -> Integer

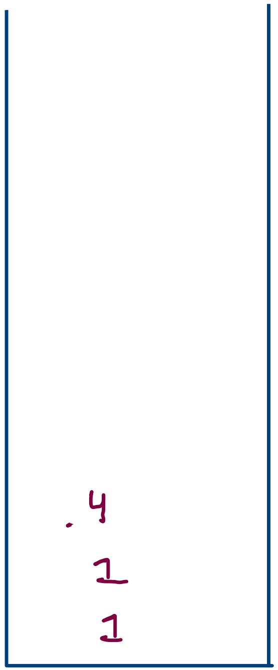
represent '(' using -1.

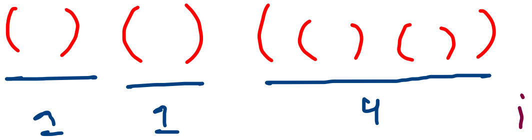


```
for(int i=0; i < s.length();i++) {
    char ch = s.charAt(i);

    if(ch == '(') {
        st.push(-1);
    }
    else {
        if(st.peek() == -1) {
            st.pop();
            st.push(1);
        }
        else {
            int sc = 0;
            while(st.peek() != -1) {
                sc += st.pop();
            }
            st.pop();
            st.push(2*sc);
        }
    }
}
```



```
int ans = 0;

while(st.size() > 0) {
    ans += st.pop();
}

return ans;
```

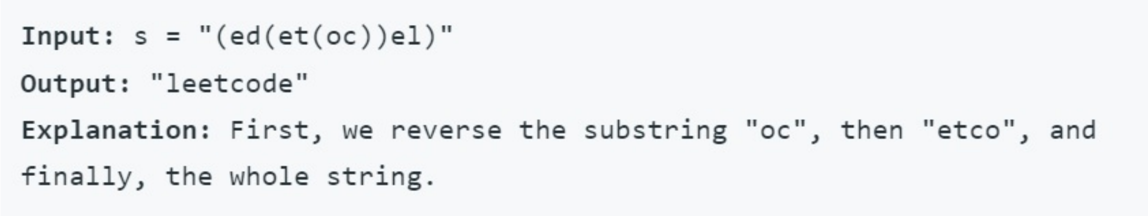




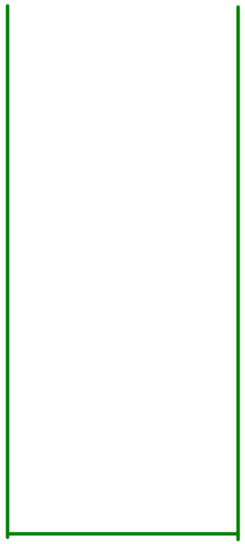
return = 4+1+1

ans = 6.

1190. Reverse Substrings Between Each Pair of Parentheses



```
Input: s = "(ed(et(oc))el)"
Output: "leetcode"
Explanation: First, we reverse the substring "oc", then "etco", and
finally, the whole string.
```



(ed (et (oc)) el) practice

(ed (et co) el) practice

(ed octe el) practice

leetcodepractice