1249. Minimum Remove to Make Valid Parentheses

15 March 2022 10:10 AM

Given a string s of `'('` , `')'` and lowercase English characters.

Your task is to remove the minimum number of parentheses (`'('` or `')'`, in any positions) so that the resulting *parentheses string* is valid and return **any** valid string.

Formally, a *parentheses string* is valid if and only if:

- It is the empty string, contains only lowercase characters, or
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are valid strings, or
- It can be written as `(A)`, where `A` is a valid string.

Example 1:

```
Input: s = "lee(t(c)o)de)"
Output: "lee(t(c)o)de"
```

```
Explanation: "lee(t(co)de)" , "lee(t(c)ode)" would
also be accepted.
```

Example 2:

```
Input: s = "a)b(c)d"
Output: "ab(c)d"
```

Example 3:

```
Input: s = "))(("
Output: ""
Explanation: An empty string is also valid.
```

S → (a (b) c) d)

o/p → (a (b) c)d or (a (b) c d) (Minimum remove 1)

↳ multiple ans

S → (a)) → (a) ✓

↳ a X (Not minimum removal)

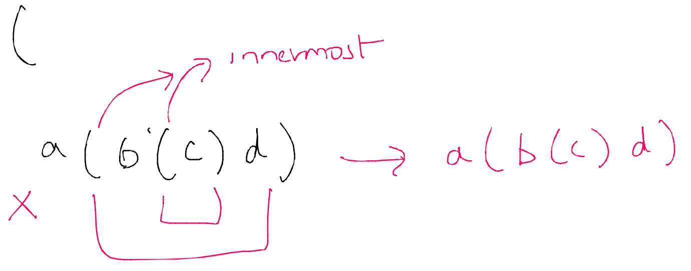
s → a((b) → a(b)

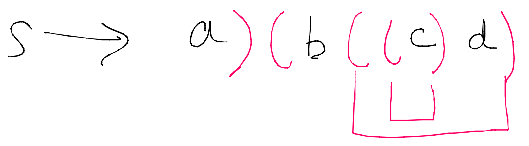
S → (a(b(c)d)

innermost (first work in innermost bracket)

Outermost

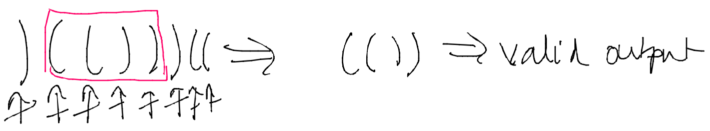
(

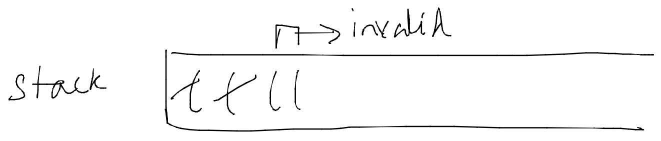




$\boxed{a\ b\ ((c)\ d)}$, ans

* give pirority to the innermost bracket





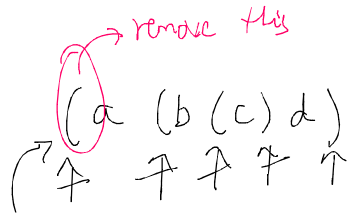
→) ⇒ no opening bracket so did'nt push that

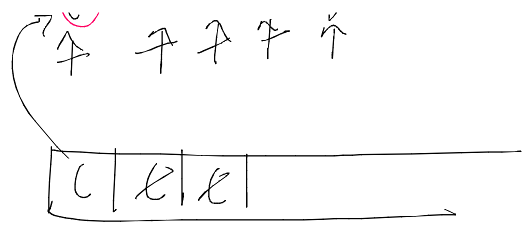
(⇒ when opening bracket comes push them

and when closing bracket comes remove the opening bracket as it make pair (it's valid)

→ Closing bracket without pair is invalid

→ Remaining Bracket in stack is also invalid.





* handle alphabet differently.

* we need to find exact position of bracket to be removed. (can be handle using array)

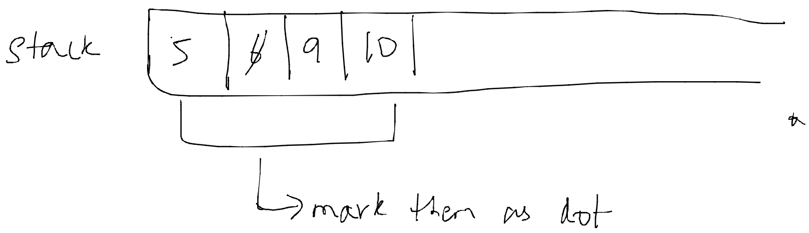
↓

(index)

0 1 2 3 4 5 6 7 8 9 10 11

a b) c d ((e) ((b

* we store the index value

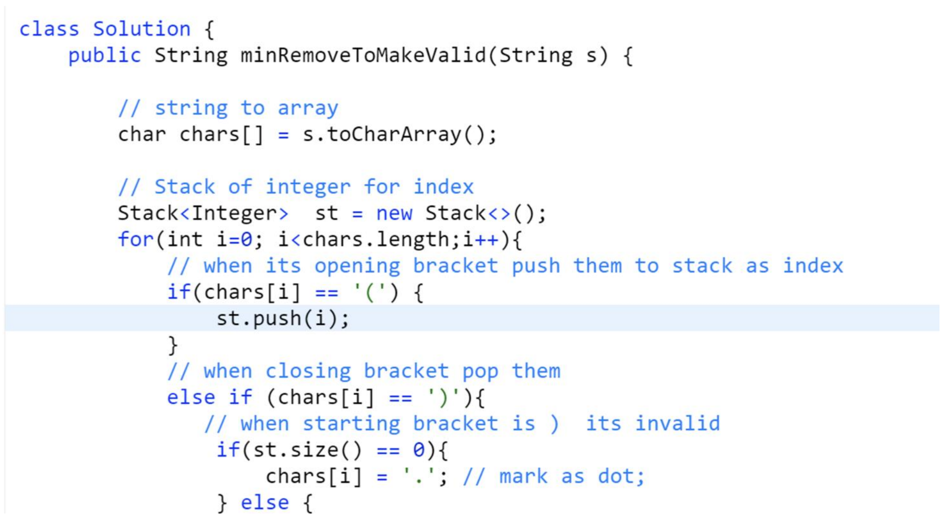


• → flag for invalid and replace the bracket with .

* in stack only opening bracket is stored

* then remove the marked dot you get the output

b c (e b ⇒ ans



```
class Solution {
    public String minRemoveToMakeValid(String s) {

        // string to array
        char chars[] = s.toCharArray();

        // Stack of integer for index
        Stack<Integer>  st = new Stack<>();
        for(int i=0; i<chars.length;i++){
            // when its opening bracket push them to stack as index
            if(chars[i] == '(') {
                st.push(i);
            }
            // when closing bracket pop them
            else if (chars[i] == ')'){
               // when starting bracket is )  its invalid
                if(st.size() == 0){
                    chars[i] = '.'; // mark as dot;
                } else {
```

```
                st.pop(); // if pair availabe pop it
            }
        }
    }
    // remaining in stack is also invalid
    while(st.size() > 0){
        chars[st.pop()] = '.';  // mark them dot
    }

    // apart from . add other to string builder as . is invalid
    StringBuilder ans = new StringBuilder();
    for(char c: chars){
        if(c != '.'){
            ans.append(c);
        }
    }
    return ans.toString();
}
}
```