Delete consecutive

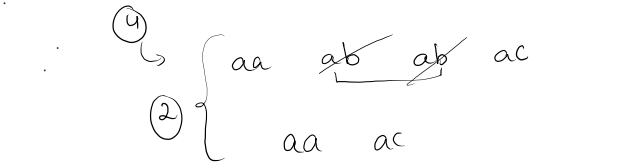
Given a sequence of N *strings*, the task is to check if any two similar words come together then they destroy each other than *print* the number of words left in the sequence after this *pairwise* destruction.

Sample Input 0

```
4
aa ab ab ac
```

Sample Output 0

```
2
```



eg2.

aa ab ab ab ac

ans = ? ⇒ 3

eg3.

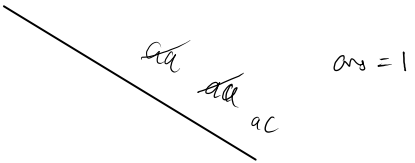
ab ab ab ab ac

ans = ? = 1

eg4.

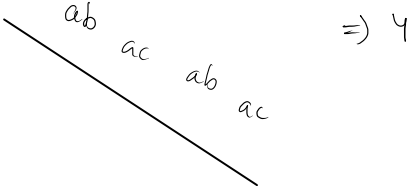
aa ab ab aa ac

ans = ? = 1



ans = 1

eg5



⇒ 4

eg. ab ac ac aa } (2)

eg 2. aa ab ab ab ac } (3)

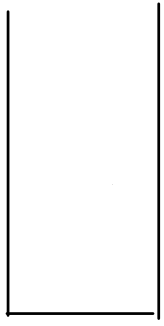
eg 3. ab ab ab ab ac } (1)

eg 4. ab ac ab ac } (4)

first or st.empty → push

if curr_ele == peek → pop

else push

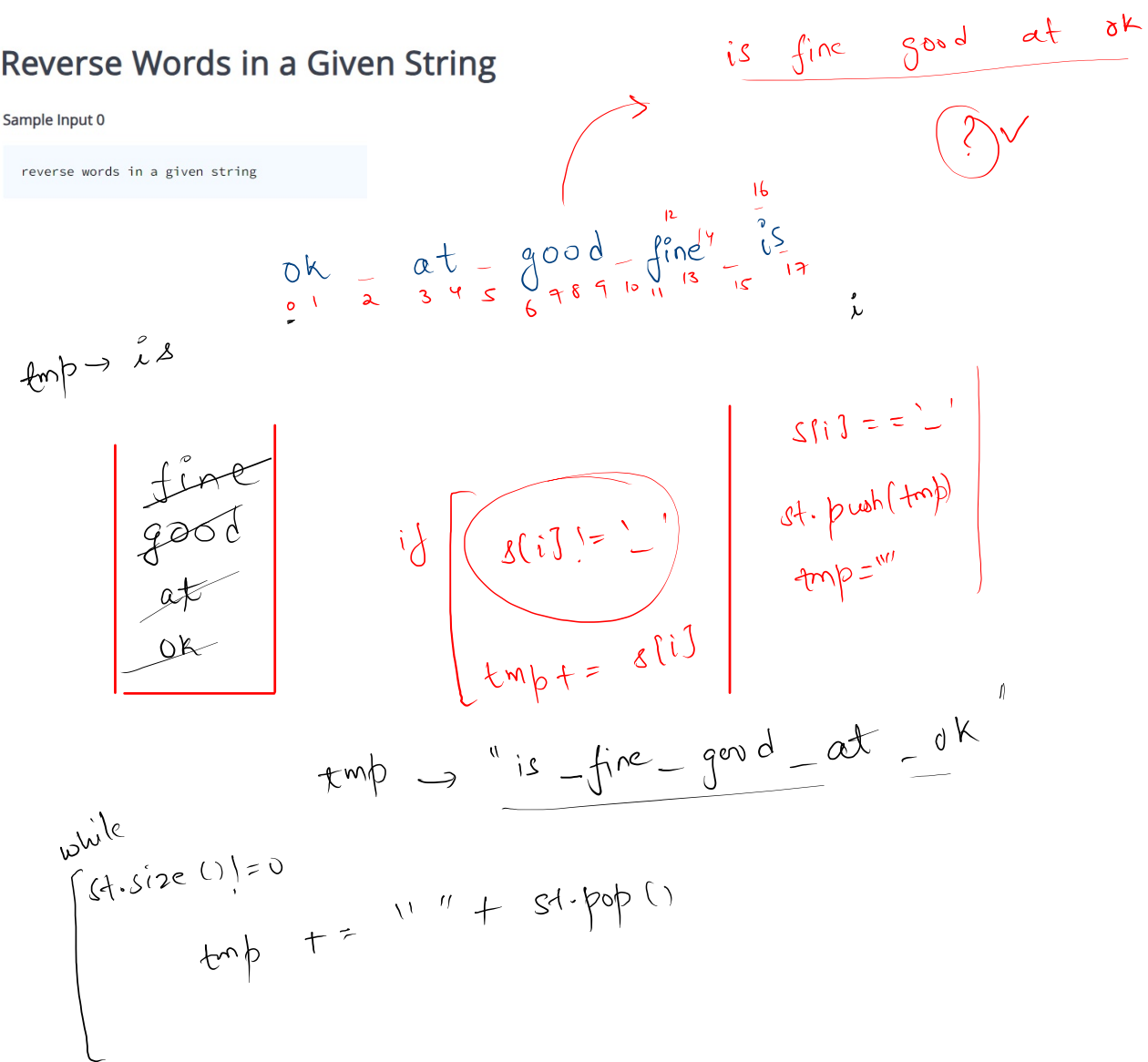


```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        Stack<String> st = new Stack<>();

        for(int i = 0; i < n; i++){
            String s = scn.next();
            if(st.size() != 0 && s.equals(st.peek())){
                st.pop();
            }else{
                st.push(s);
            }
        }
        System.out.println(st.size());
    }
}
```

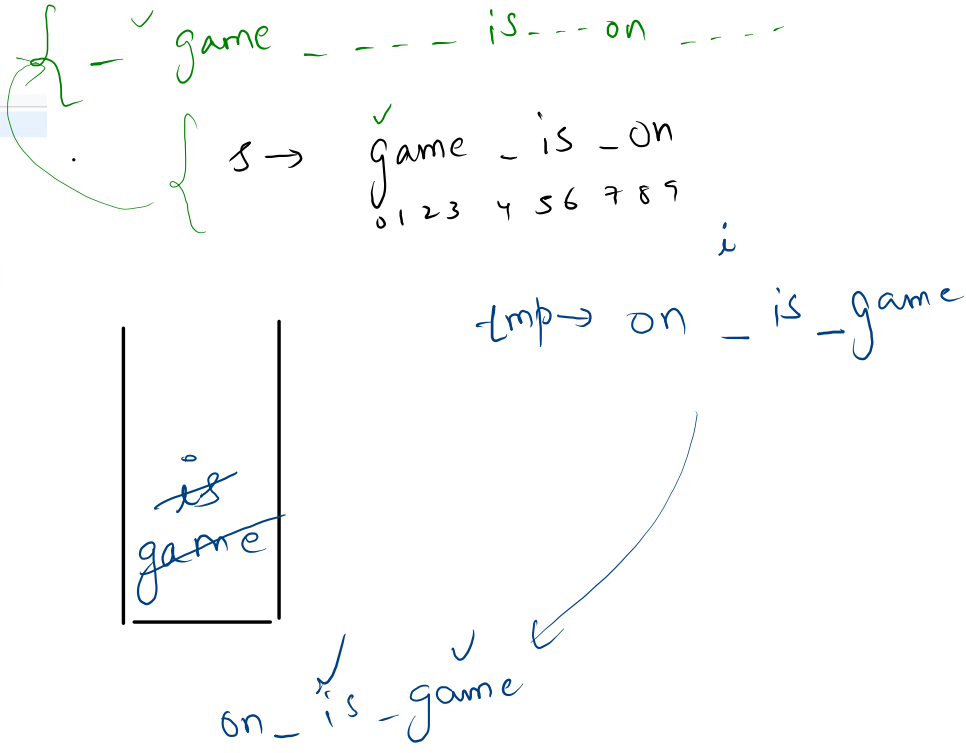
Reverse Words in a Given String

Sample Input 0

```
reverse words in a given string
```



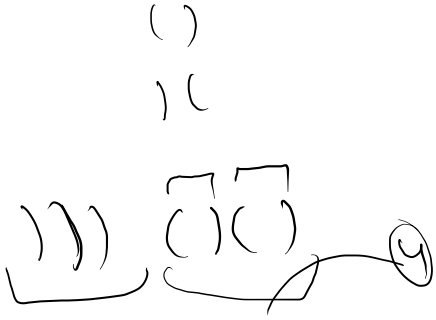
```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.nextLine();
        Stack<String> st = new Stack<>();

        String tmp = "";
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            if(ch != ' '){
                tmp += ch;
            }else{
                st.push(tmp);
                tmp = "";
            }
        }
        //remove
        while(st.size() != 0){
            tmp += " " + st.pop();
        }


        System.out.println(tmp);
    }
}
```

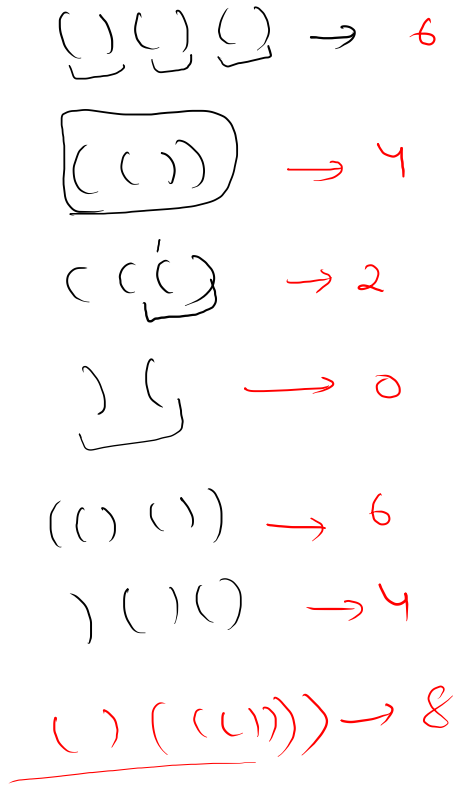


Longest Valid Parentheses 4

Problem | Submissions | Leaderboard | Discussions

Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring.





8 → ( ) (( ) ( )
0 1 2 3 4 5 6

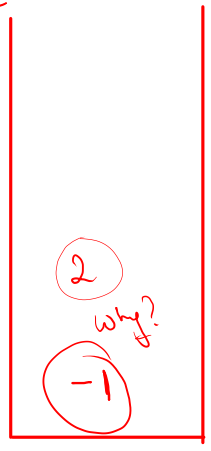
ans = 0 2 4

i

if → s[i] == '(' → push(i)

↳ ')' → st.pop() } to calc. length
find len

idx



len = i - peek

6 - 2 = 4

s →) (( )) (( )

0 1 2 3 4 5 6 7

i

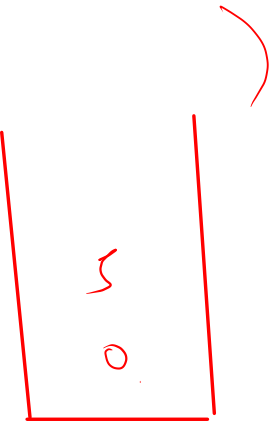
ans = ~~0~~ ~~2~~ 4

s[i] == '(' → push

s[i] == ')' → remove

len

len = i - peek



len = 2

len = 4

len = 2

```java
class Solution {
    public int longestValidParentheses(String s) {
        Stack<Integer> st = new Stack<>();
        st.push(-1);
        int ans = 0;
        int n = s.length();
        for(int i = 0; i < n; i++){
            char ch = s.charAt(i);
            if(ch == '('){
                st.push(i);
            }else{
                //remove and find len
                st.pop();
                if(st.size() == 0){
                    st.push(i);
                }else{
                    int len = i - st.peek();
                    ans = Math.max(ans, len);
                }
            }
        }
        return ans;
    }
}
```