LIFO

push

pop

peek

size

Reverse string

Problem | Submissions | Leaderboard | Discussions

Given a String Str. We have to $Reverse$ the string Str with help of only $stacks$.

Sample Input 0

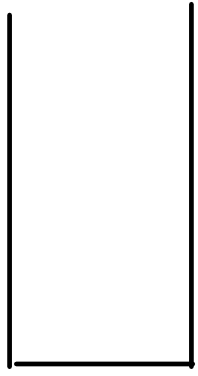
```
abcdee
```

Sample Output 0

```
eedcba
```

abc dee → eedcba

steps.



eedcba

```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.next();

        Stack<Character> st = new Stack<>();
        for(int i = 0; i < s.length(); i++){
            st.push(s.charAt(i));
        }
        String ans = "";
        while(st.size() != 0){
            ans += st.pop();
        }
        System.out.println(ans);
    }
}
```

s → a b d e e

0 1 2 3 4

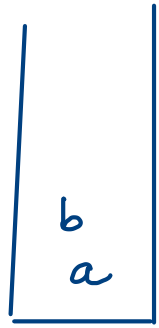
i=0

0<5

5!=0

4!=0

3!=0



st

ans → e e d b a

ans → e e d

Delete consecutive

Problem | Submissions | Leaderboard | Discussions

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
```

aa ab ab ac

2

eg. 5

aa bc ab ab bc ac ?

1. aa ~~ab~~ ~~ab~~ ac → 2 ✓

2. aa ~~bc~~ ~~ab~~ ~~ab~~ ~~bc~~ ac → 2

3. aa ab bc ab ac → 5

4. ~~ab~~ ~~ab~~ ab → 1

5. bc ~~ab~~ ~~ab~~ ab ac → 3.

Logic.

aa ab bc bc ab ac → 2

ac

aa

2 - 3 mins.

code

aa aa aa ab ab ac } → 2

```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();// 6
        Stack<String> st = new Stack<>();
        for(int i = 0; i < n; i++){        //n times
            String s = scn.next();         // aa ab ab ac
            if(st.size() != 0 && st.peek().equals(s)){
                st.pop();
            }
            else{
                st.push(s);
            }
        }
        System.out.println(st.size());
    }
}
```

aa ac

Reverse Words in a Given String

Problem | Submissions | Leaderboard | Discussions

Mr. Reverse can only read reverse sentences.

If a sentence is : ***i like geekster***

Mr. Reverse will be able to read it only if it is written as: ***geekster like i***

Let's help him read.

Write a program to reverse the words in a given sentence.

Take a **String str** as input and **Reverse** all the words in a given string.

Sample Input 0

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

Sample Output 0

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

reverse_words_in_a_given_string

tmp = in

string
given
a
in
words
reverse

```java
import java.util.*;
public class Main
{
    public static void main(String[] args) {
        String s = "Aman geekster open close";
        String [] A = s.split(" ");       // {"Aman", "geekster", "open", "close"}

        for(String w : A){
            System.out.println(w);
        }

        System.out.println(A.length);
    }
}
```

close
open
geekster
Aman

close _ open _ geekster _ Aman

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.nextLine();
        Stack<String> st = new Stack<>();
        String tmp = "";
        for(int i = 0; i < str.length(); i++){
            if(str.charAt(i) == ' '){
                st.push(tmp);
                tmp = "";
            }
            else{
                tmp += str.charAt(i);
            }
        }
        st.push(tmp);
        String ans = "";
        while(st.size() != 0){
            ans += st.pop() +" ";
        }
        System.out.println(ans);
    }
}
```

str → hello_super_human

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

tmp = human

ans = (human_ super_hello_)

human
super
hello

aman _ _ geeks _ _ .

_ _ . aman _ _ _ geeks _ _ _

Valid Parentheses 4

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.

Sample Input 0

```
()[]{}
```

Sample Output 0

```
true
```

eg. [ ] → true

[( )] → true

[(( )] → false

( ) [ ] ( )
↳ true

) (→ false.

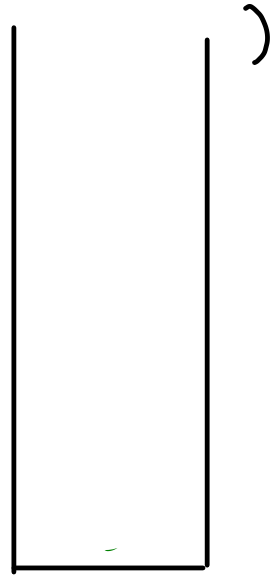
(→ false.

( ) [[ ]] { } → true

open

({) → false

) (→ false



```java
        boolean ans = true;

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

            if(ch == '{' || ch == '(' || ch == '['){//open bracket
                st.push(ch);
            }
            else if(st.size() == 0){
                ans = false;
                break;
            }
            else if(ch == '}' && st.peek() != '{'){
                ans = false;
                break;
            }
            else if(ch == ']' && st.peek() != '['){
                ans = false;
                break;
            }
            else if(ch == ')' && st.peek() != '('){
                ans = false;
                break;
            }
            else{
                st.pop();
            }
        }
        if(st.size() != 0){
            ans = false;
        }

        System.out.println(ans);
```

```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.next();

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

        boolean ans = true;

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

            if(ch == '{' || ch == '(' || ch == '['){//open bracket
                st.push(ch);
            }
            else if(st.size() == 0){
                ans = false;
                break;
            }
            else if(ch == '}' && st.peek() != '{'){
                ans = false;
                break;
            }
            else if(ch == ']' && st.peek() != '['){
                ans = false;
                break;
            }
            else if(ch == ')' && st.peek() != '('){
                ans = false;
                break;
            }
            else{
                st.pop();
            }
        }
        if(st.size() != 0){
            ans = false;
        }

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