


Stack

LIFO \rightarrow last in, first out
semantics

\Rightarrow Syntax

`Stack < Integer > st = new Stack < > ();`

\Rightarrow function

1) add) `st.push(5)`

2) remove) `int a = st.pop()` \rightarrow remove and return the top element

3) `int a = st.peek()` \rightarrow returns the top element

4) size) `st.size()`

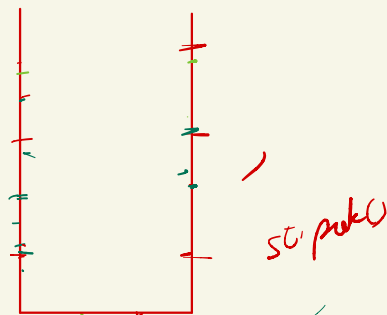
```
public class Example {  
    Run | Debug  
    public static void main(String[] args) {  
        Stack<Integer> st = new Stack<>();  
  
        st.push(item: 1);  
        st.push(item: 2);  
        st.push(item: 3);  
  
        System.out.println(st.peek());  
  
        st.push(item: 4);  
        st.push(item: 5);  
  
        while(st.size() > 0){  
            int a = st.pop();  
            System.out.println(a);  
        }  
    }  
}
```



`int a = st.pop()`

5
4
 \hookrightarrow 3

2
1



int $a = 5;$

$a = 5$ return job

```
public boolean isValid(String s) {
    Stack<Character> st=new Stack<>();

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

        if(ch=='(' || ch=='{' || ch=='['){
            st.push(ch);
        } else if(ch==')'){
            if(st.size()==0 || st.peek()!='(') return false;

            st.pop();
        } else if(ch=='}'){
            if(st.size()==0 || st.peek()!='{') return false;

            st.pop();
        } else if(ch==']'){
            if(st.size()==0 || st.peek()!='[') return false;

            st.pop();
        }
    }

    if(st.size()==0) return true;

    return false;
}
```

Ques input string ()


$$a = \cancel{2} / \cancel{2} / \cancel{2} / 0$$

(()) (())

$$a = 8 + 2 + 8 + 2 + 2 = 25$$

C))C

$O(1)$ space
 $O(N)$ time

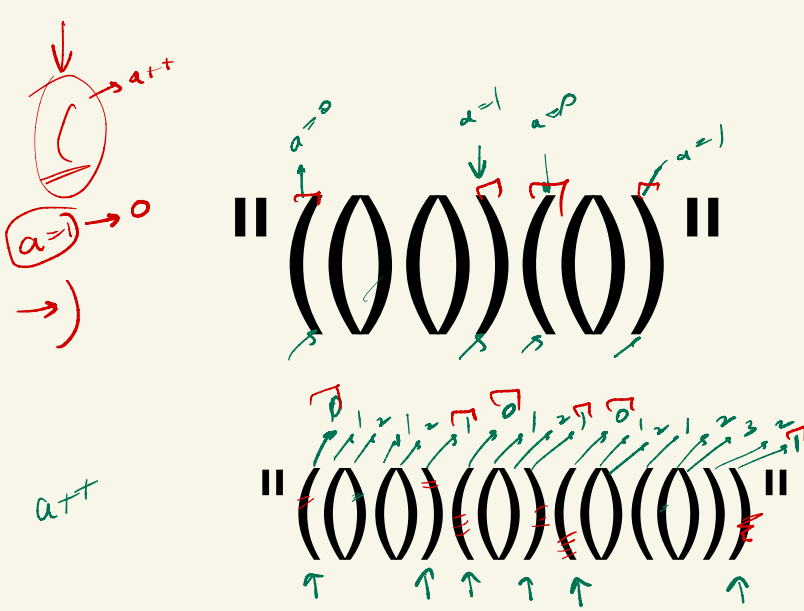
'y' ($a < 0$)
return false

```
if (ch == 'c') a++;
else a--;
```

- y(a == 0)
return true

$a = 1$

$(())(())$
 $a = 4 \times 2 \times 2 \times 1$



a
↳ number of opening brackets with no corresponding closing bracket

if (ch == '(' and a == 0)
if (ch == ')' and a == -1)

Next greater

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

```

// Next greater on right
public static long[] nextLargerElement(long[] arr, int n){
    long[] ngr = new long[n];

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

    for(int i=n-1; i>=0; i--){
        long ele = arr[i];

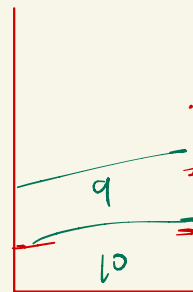
        while(st.size()>0 && st.peek()<=ele){
            st.pop();
        }

        if(st.size()==0){
            ngr[i]=-1;
        } else {
            ngr[i]=st.peek();
        }

        st.push(ele);
    }

    return ngr;
}

```



ng(a)
arr(st.peek())