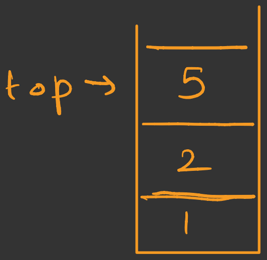
# STACK

↳ Linear Data Structure

↳ LIFO (Last In First Out)



eg 1) Function / Call stack

2) Undo / Redo

3) Forward / Back in a browser

## Functions of Stack

① Push → add at top

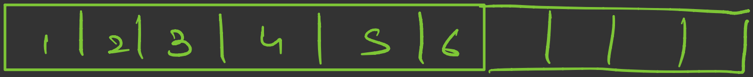
② Pop → remove top

③ Peek → returns top

④ size → return size of stack

⑤ IsEmpty → returns s==0;

ArrayList



addFirst O(n) removeFirst O(n)

addLast O(1) removeLast O(1)

```
class Stack {
 ArrayList<Integer> list;

 p void  push(int val) {
     list.addLast(val);
 }
 p  int  pop() {
    return list.removeLast();

 }
 p int  peek() {
   return list.get(list.size()-1)
 }
```

list

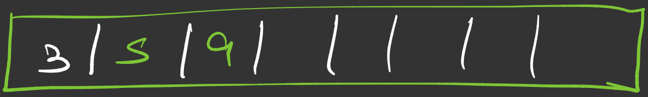
Linked List

add First O(1)    remove First O(1)

add Last O(n)    remove Last O(n)

if tail O(1)

Array



```
Class Stack {

    int arr[];

    int top;   // -1

p   void push(int val) {

        arr[++top] = val;

    }
```

push(3)
push(5)
push(9)
pop()
peek()

```
p  int pop() {
     return arr[top--];
}
```

size()
return top+1;

## Ques Duplicate brackets

((a + b) + c) // false

↑
i

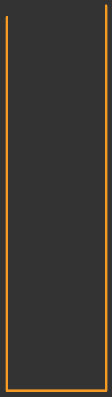
(((a + b)) + c) // true

---

## Ques Balanced Bracket

{a + (b + c) + [(d + e) + f]} // true

↑



{ ( ] }

{ ( } )

} {

[ { ] }

false

/ * + − %   Operators

A , B , C . . . .   Operands

$\rightarrow ((A + B) - ((C + D) / E))$ ] Infix Expression

$+AB - (+CD / E)$

$+AB - (/+CDE)$

$- +AB /+CDE$ ] Prefix Expression

---

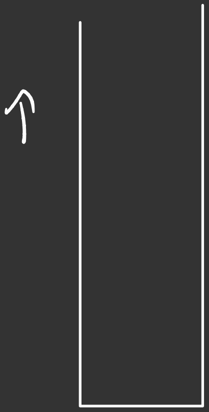
<u>Ques</u> Prefix to Infix

$\uparrow$ $-+AB/+CDE$

$((A + B) - ((C+D) / E))$

**Ques** Nearest smaller element (on left)

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 4 | 5 | 6 | 10 | 8 | 5 | 1 |
| -1 | 4 | 5 | 6 | 6 | 4 | -1 |



TC ⇒ O(n)