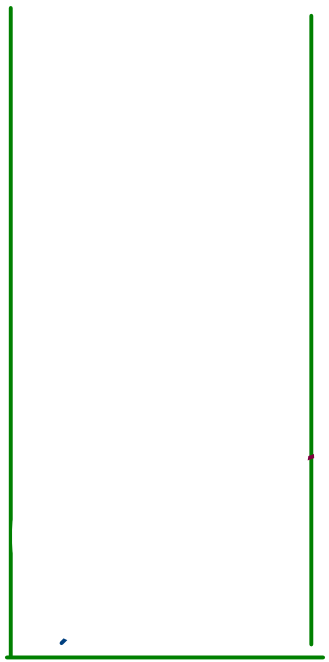
Duplicate brackets →

string →

e.g.'
((a + b) + (c + d)) -> false
(a + b) + ((c + d)) -> true

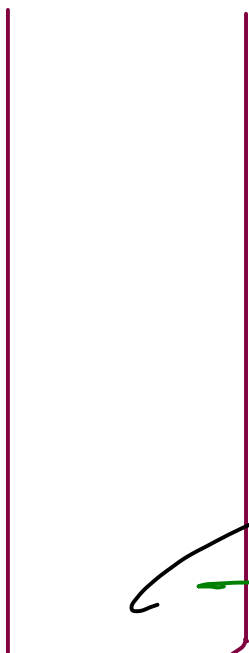


for(int i = 0; i < st

Balanced Bracket →

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

```
[(a + b) + {(c + d) * (e / f)}] -> true
[(a + b) + {(c + d) * (e / f)]} -> false
[(a + b) + {(c + d) * (e / f)} -> false
([(a + b) + {(c + d) * (e / f)}] -> false
```



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

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

Return false

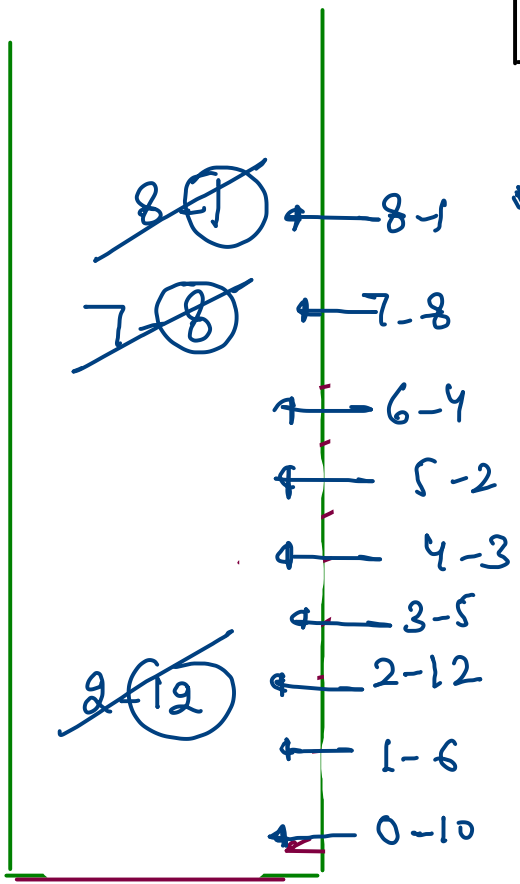
{( ]}

Next Greater on Right

Index

0	1	2	3	4	5	6	7	8
10	6	12	5	3	2	4	8	1

0	1	2	3	4	5	6	7	8
12	12	-1	8	4	4	8	-1	-1



time $O(n)$ →

space → $O(n)$

pop - $O(1)$

push - $O(1)$

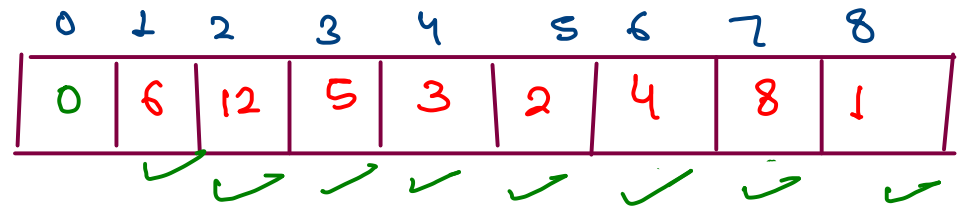
Steps
1. Make an array of size 'n'
2. make Stack, push '0' th index
3. start from i = 1 &
4. maintain the decreasing order of val from stack & arrays

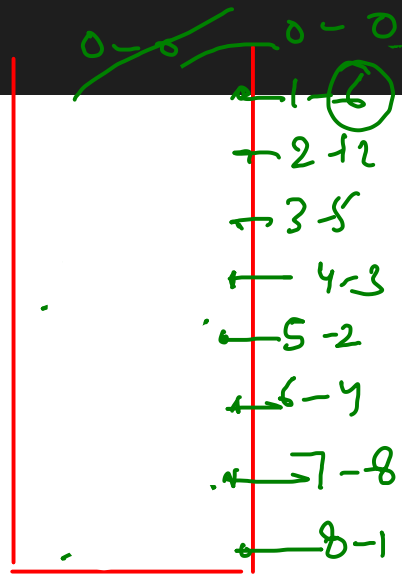
Algo

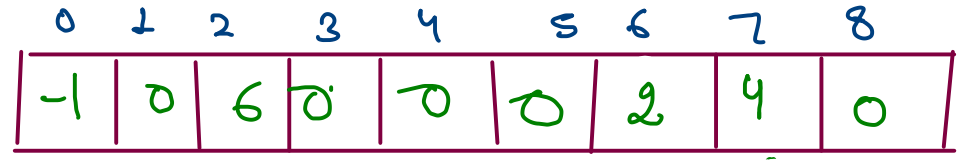
```java
// nsl -> next smaller on left
public static int[] nsl(int[] arr) {
    int[] res = new int[arr.length];
    Stack<Integer> st = new Stack<>();
    st.push(arr.length - 1);

    for(int i = arr.length - 2; i >= 0; i--) {
        while(st.size() > 0 && arr[st.peek()] > arr[i]) {
            res[st.pop()] = arr[i];
        }
        st.push(i);
    }
    while(st.size() > 0) {
        res[st.pop()] = -1;
    }
    return res;
}
```

```
arr : [10, 6, 12, 5, 3, 2, 4, 8, 1]
ngr : [12, 12, -1, 8, 4, 4, 8, -1, -1]
ngl : [-1, 10, -1, 12, 5, 3, 5, 12, 8]
nsr : [6, 5, 5, 3, 2, 1, 1, 1, -1]
nsl : [-1, -1, 6, -1, -1, -1, 2, 4, -1]
```







Stock Span

next greater on left.

Days	0	1	2	3	4	5	6	7	8
Price →	10	6	12	5	3	2	4	8	1

	0	1	2	3	4	5	6	7	8
	1	1	3	1	1	1	3	5	1

Span of Stock

↓

How many days before today, stock price is Higher than today.

y [2 5 9 3 1 1 2 6 8 7]

0 1 2 3 4 5 6 7 8

index	0	1	2	3	4	5	6	7	8
of next greater on left →	-1	-1	-1	2	3	-1	5	5	7

0 - (-1)

	1	2	3	1	1	6	1	2	1

span for 2 is 1
span for 5 is 2
span for 9 is 3
span for 3 is 1
span for 1 is 1
span for 12 is 6
span for 6 is 1
span for 8 is 2
span for 7 is 1