Stack

↳ LIFO (Last In First Out)

=> Stack < Datatype > stack_name = new Stack <>();

→ sf.push( )

→ sf.pop()

→ sf.size()

→ sf.peek()

① (a + b) + ((c + d))

↑

↳ !) → st.push.

↳) → st.peek() = '(' → extra

↳ pop !! → st.peek() = '(' →

st.pop()

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

Balanced Bracket (Day 34)

Problem | Submissions | Leaderboard | Discussions

1. You are given a string exp representing an expression.
2. You are required to check if the expression is balanced i.e. closing brackets and opening brackets match up well.

Sample Input 0

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

Sample Output 0

```
true
```

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

exp → [[(a+b) + (c+d)] }] → False

→ [, (, {

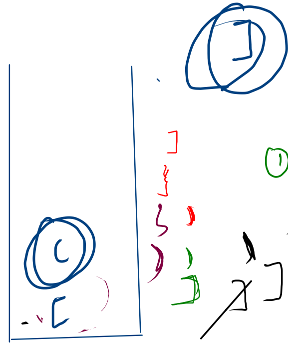
[(a+b) + { (e+d) * (e / f) }]

① → Opening bracket :- push into the stack.

② → Ignore characters other than brackets.

③ → Closing → (←), { }, [ ] → pop the opening bracket.

if (st.size() == 0) → true.
else → false;



False

① Mis match of bracket → False

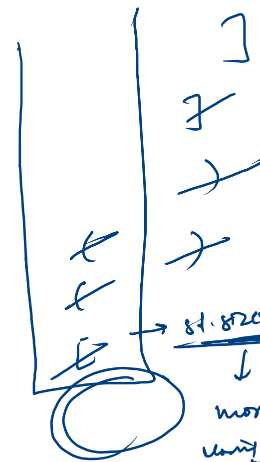
② when closing bracket > opening bracket.

③ when opening bracket > closing bracket

Ex-① [(a+b) * (c+d) * { e+f] } → Mismatch.

ex ② [(a+b) + [e+d)]] → more closing brackets.

ex ③ [{ { a+b } + (c+d) } → more opening brackets.



st.size() == 0 → more closing brackets

st.pop() → error

```java
public static boolean balanceBracket(String str){
    Stack<Character> st = new Stack<>();

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

        if(ch == '(' || ch == '{' || ch == '['){
            st.push(ch);
        }else if(ch == ')' || ch == ']' || ch == '}'){
            if(st.size() == 0) return false; // more closing brackets

            // Line no. 17 - 28 => Mismatch of brackets
            else if(ch == ')'){
                if(st.peek() != '(') return false;
            }

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

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

            st.pop();
        }
    }

    if(st.size() == 0) return true;
    return false; // more opening brackets
}
```

Next Greater Element To The Right (Day 35)

Problem | Submissions | Leaderboard | Discussions

1. You are given a number n, representing the size of array a.
2. You are given n numbers, representing elements of array a.
3. You are required to "next greater element on the right" for all elements of array
4. Input and output is handled for you.

"Next greater element on the right" of an element x is defined as the first element to right of x having value greater than x. Note -> If an element does not have any element on it's right side greater than it, consider -1 as it's "next greater element on right" e.g. for the array [2 5 9 3 1 12 6 8 7] Next greater for 2 is 5 Next greater for 5 is 9 Next greater for 9 is 12 Next greater for 3 is 12 Next greater for 1 is 12 Next greater for 12 is -1 Next greater for 6 is 8 Next greater for 8 is -1 Next greater for 7 is -1

Sample Input 0

```
5
5
3
8
-2
7
```

Sample Output 0

```
8
8
-1
7
-1
```

n = 5

arr → | 2 | 5 | 9 | 3 | 1 | 12 | 6 | 8 | 7 |

nge → | 5 | 9 | 12 | 12 | 12 | -1 | 8 | -1 | -1 |

arr → | 5 | 3 | 8 | -2 | 7 |

nge → | 8 | 8 | -1 | 7 | -1 |

```
for (int i = 0; i < n; i++)
{
    for (int j = i+1; j < n; j++) {
        if (arr[i] < arr[j]) → nge[i] = arr[j]; break;
    }
}
```

T.C O(n^2)

i = 0 1-n, i = 1 2-n, i = 2 3-n, i = 3 4-n → n^2

i = n-2; i >= 0; i--

st.push(arr[n-1])

i = 3 >= 0, st.peek() > arr[i]), 7 > -2

i = 2 >= 0, -2 > 8, 7 > 8

i = 1 → 8 > 3

i = 0 → 5, 8 > 5

j = -1 >= 0 ✗

st.size() == 0 → element greater than the arr[i] exists on the right side

st.peek() > arr[i]

i = 7 → 7 > 8 (F)

i = 6 → 8 > 6 (T)

i = 5 → 6 > 12 (F), 8 > 12 (F)

i = 4 → 12 > 1 (T)

i = 3 → 1 > 3 (F), 12 > 3 (T)

i = 2 → 3 > 9 (F), 12 > 9 (T)

i = 1 → 9 > 5 (T)

i = 0 → 5 > 2 (T)

i = -1 >= 0 (F)

T.C = O(n)

S.C = O(n)

```java
public static void ngr(int arr[],int n){
    Stack<Integer> st = new Stack<>();
    int nge [] = new int [n];
    nge[n-1] = -1;

    st.push(arr[n-1]);

    for(int i=n-2;i>=0;i--){
        while(st.size() > 0  && st.peek() < arr[i]){
            st.pop();
        }

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

        st.push(arr[i]);
    }

    for(int i=0;i<n;i++){
        System.out.println(nge[i]);
    }
}

public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print output
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int arr[] = new int[n];
    for(int i=0;i<n;i++){
        arr[i] = scn.nextInt();
    }

    ngr(arr,n);
}
```

↳ $O(n^2)$

Why? ⇒ because

T.C. ⇒ $O(n)$

10 mins ⇒ 5 Coding + 5 Dry Run

Next Greater Element To The Left (Day 35)

Problem | Submissions | Leaderboard | Discussions

1. You are given a number n, representing the size of array a.
2. You are given n numbers, representing elements of array a.
3. You are required to find "next greater element on the left" for all elements of array
4. Input and output is handled for you.

Sample Input 0

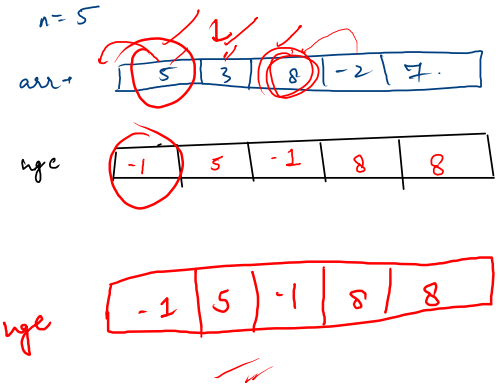
```
5
5
3
8
-2
7
```

Sample Output 0

```
-1
5
-1
8
8
```

nge on right → `r = n-2; i>=0; r--`

nge on left → `i=1; i<n; i++`



8

st.peek() > arr[i]

5 > 3

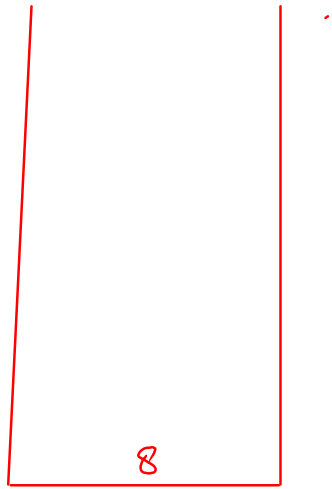
3 > 8

5 > 8

8 > -2

-2 > 7

8 > 7



```java
public class Solution {
    public static void ngl(int arr[],int n){
        Stack<Integer> st = new Stack<>();
        int nge[]= new int[n];
        nge[0] = -1;
        st.push(arr[0]);

        for(int i=1;i<n;i++){
            while(st.size() > 0 && st.peek()<arr[i]) st.pop();

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


            st.push(arr[i]);
        }

        for(int i=0;i<n;i++){
            System.out.println(nge[i]);
        }
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr[] = new int [n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        ngl(arr,n);
    }
}
```