Stacks and Queues

✅ 1. Implement a Stack Using List

Description: Implement basic stack operations: `push`, `pop`, and `peek`.

Input:

```
s = Stack()
s.push(10)
s.push(20)
s.push(30)
print(s.pop())
print(s.peek())
```

Output:

```
30
20
```

✅ 2. Balanced Parentheses

Description: Given a string with brackets, check if it is balanced.

Input:

```
s = "{[()()]}"
```

Output:

True

✅ 3. Next Greater Element

Description: For every element in the array, find the next greater element to its right.

Input:

arr = [4, 5, 2, 25]

Output:

[5, 25, 25, -1]

✅ 4. Reverse a Queue

Description: Reverse the elements of a queue using a stack.

Input:

q = [1, 2, 3, 4, 5]

Output:

[5, 4, 3, 2, 1]

✅ 5. Implement a Queue Using Two Stacks

Description: Implement `enqueue` and `dequeue` using two stacks.

Input:

```
q = MyQueue()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
print(q.dequeue())
print(q.dequeue())
```

Output:

1
2

✅ 6. Sort a Stack

Description: Sort the elements of a stack using another stack (no other data structures).

Input:

stack = [34, 3, 31, 98, 92, 23]

Output:

[3, 23, 31, 34, 92, 98]

✅ 7. Sliding Window Maximum (Queue-based)

Description: Given an array and a window size k, print the max of each subarray of size k.

Input:

nums = [1,3,-1,-3,5,3,6,7]
k = 3

Output:

[3, 3, 5, 5, 6, 7]

✅ 8. Evaluate Reverse Polish Notation (Postfix)

Description: Evaluate an expression given in postfix notation.

Input:

tokens = ["2", "1", "+", "3", "*"]

Output:

9

✅ 9. Celebrity Problem

Description: In a party of N people, a celebrity is known by everyone but knows no one. Find the celebrity using a stack.

Input:

```
M = [
 [0, 1, 1],
 [0, 0, 1],
 [0, 0, 0]
]
```

Output:

2

✅ 10. Queue with Get Min() in O(1)

Description: Implement a queue that supports `getMin()` in constant time.

Input:

```
q = MinQueue()
q.enqueue(3)
q.enqueue(1)
q.enqueue(2)
print(q.getMin())  # 1
q.dequeue()
print(q.getMin())  # 1
q.dequeue()
print(q.getMin())  # 2
```

Output:

1
1
2