STACK PUSH AND POP: These operations depend on which data structure is used to implement the stack. For example, as stack implemented using an array would have a worst case complexity of O(n) for it's push operation, as the array will potentially require resizing, a O(n) operation. However, if the stack is implemented using a linked or doubly linked list, both the push and pop operation can be done in constant time complexity, O(1), as only one memory address will need to be updated to add or remove a node.

STACK PUSH W/ QUEUE: Once again, the time complexity of these operations could vary depending on how they are implemented, but I will only evaluate for a linked list here as that is what I used. For the worst case of a push operation using queues, the entire queue would need to be dequeued and requeued. While the individual enqueue and dequeue operations run in constant time with a linked list, performing these operations for every item gives a worst case complexity of O(n)

STACK POP W/ QUEUE: For a linked list approach, pop operation only requires a single dequeue operation, meaning this only needs to update a couple of memory addresses, and giving a worst case complexity of O(1).