Tyler Forrester
Assignment Fa: Test Cases

A. Testing Questions

Why do you only need a singly linked list to implement the stack structure?

By using two pointers to keep track of the structure, one for the first node, *head* and one for the last node, *back*. We can find any node in the stack structure. Since each node is connect by a pointer to the next node in the list we simply need to start at the head and then move toward the back resetting the next pointers as they go. If we need to *peek* the last node in, we can use the back pointer. If we need to *pop*, the last node, we'll iterate through the list to find the node right behind the last node. Once we've found this node, we can easily delete the back node and set the current back node to the previously previous node, which has been promoted to the last node in the list. We only need to traverse the node in one direction to do all of these steps, thusly, we only need a singly linked list to accomplish the stack function.

How can you tell that the stack and queue structures are working correctly?

First to use the peek function to return the values from the stack in a last in-first out order. Then use the synonymous function on the queue structure making sure its first in first out. We should of course then test to make sure the structure is empty when after the list of integers has been displayed. Ideally this will be in the form of printout. Once we've established these structures work for the general cases say, 8 – 16 elements, then we should also check the special cases where the structures contain 1 or 2 items using the same aforementioned tests. By design, the program can only have greater than one node in the display list function. If they pass all of these tests, then we know that the structures are working correctly.

What happens when I deliberately make the queue and stack functions remove too many items. Why do you not need to test for this?

A segmentation fault occurs as the program tries to access memory that is not allocated to it. After modifying my return functions to catch the error. The program now returns 0 with a Printout that says "Tried to Print Out an Undefined Element". We don't need to test for this outcome because our user is never (outside of the demonstration mode) allowed to remove individual items from the list. The user list is always displayed after entry and the user is never allowed to call a blank list in the main function. The while loop logic which prints values until all the structures are removed. Once the structures are removed then the program ceases to print values.

B. Testing Plan

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Add 6 Integers	1,n,2,n,3,n,4,n,5,n,6, q	Main()menu()getchoice() stack.isEmpty() stack.push() stack.pop() stack.peek() queue.addBack() queue.removeFront() queue.getFront() queue.isEmpty()	Please enter an integer: 1 n Please enter an integer: 2 n Please enter an integer: 3 n Please enter an integer: 4 n Please enter an integer: 5 n Please enter an integer: 6 6 Your List has been entered. Please enter 'c' to continue. c Displaying your stack structure 6 5 4 3 2 1 Please enter 'c' to continue. c Displaying your queue structure 1 2 3 4 5 6 Please enter 'c' to continue. c There are no numbers left in your Stack list There are no numbers left in your Queue list	Please enter an integer: 1 n Please enter an integer: 2 n Please enter an integer: 3 n Please enter an integer: 4 n Please enter an integer: 5 n Please enter an integer: 6 6 Your List has been entered. Please enter 'c' to continue. c Displaying your stack structure 6 5 4 3 2 1 Please enter 'c' to continue. c Displaying your queue structure 1 2 3 4 5 6 Please enter 'c' to continue. c There are no numbers left in your Stack list There are no numbers left in your Queue list
Add 2 Integers	1,n,2,q	Main()menu()getchoice() stack.isEmpty() stack.push() stack.pop()	Please enter an integer: 1 n	Please enter an integer: 1 n

		stack.peek() queue.addBack() queue.removeFront() queue.getFront() queue.isEmpty()	Please enter an integer: 2 n Your List has been entered. Please enter 'c' to continue. c Displaying your stack structure 2 1 Please enter 'c' to continue. c Displaying your queue structure 1 2 Please enter 'c' to continue. c There are no numbers left in your Stack list There are no numbers left in your Queue list	Please enter an integer: 2 n Your List has been entered. Please enter 'c' to continue. c Displaying your stack structure 2 1 Please enter 'c' to continue. c Displaying your queue structure 1 2 Please enter 'c' to continue. c There are no numbers left in your Stack list There are no numbers left in your Queue list
Add 1 Integers	1, q	Main()menu()getchoice() stack.isEmpty() stack.push() stack.pop() stack.peek() queue.addBack() queue.removeFront() queue.getFront() queue.isEmpty()	Please enter an integer: 1 n Your List has been entered. Please enter 'c' to continue. c Displaying your stack structure 1 Please enter 'c' to continue. c Displaying your queue structure 1 Please enter 'c' to continue. c	Please enter an integer: 1 n Your List has been entered. Please enter 'c' to continue. c Displaying your stack structure 1 Please enter 'c' to continue. c Displaying your queue structure 1 Please enter 'c' to continue. c

			There are no numbers left in your Stack list There are no numbers left in your Queue list	There are no numbers left in your Stack list There are no numbers left in your Queue list
2 Try to print out an Undefined Node	Choice = 2	Menu() getchoice() queue.getFront(), stack.peek()	Attempt to print out N+1 values where N = 0. Tried to Print Out an Undefined Element This is the queue 0 Tried to Print Out an Undefined Element This is the stack 0	Attempt to print out N+1 values where N = 0. Tried to Print Out an Undefined Element This is the queue 0 Tried to Print Out an Undefined Element This is the stack 0
3: Exit Program	Choice = 3	Main()	Successfully Exited Program	Successfully Exited Program