Performance:

In stack, the push, pop, and peek methods are all constant time performance. Since we store the location of the front in both array deque and linked list deque, albeit in different ways, we can access and perform operations on the front in constant time, which is all we need in a stack. This means that it will take the same number of steps to push, pop, or peek no matter the size of the stack. Len is also constant time, as it just returns a value. Str is linear time because it returns the value returned by the str method of either array deque or LL deque, which both run in linear time, as the number of steps increases as the input increases due to the presence of a loop.

In queue, the enqueue, dequeue, and peek methods are all constant time performance. Since we store the location of the front in both array deque and linked list deque, albeit in different ways, we can access and perform operations on the front and back in constant time, which is all we need in a queue. This means that it will take the same number of steps to enqueue, dequeue, or peek no matter the size of the queue. Len is also constant time, as it just returns a value. Str is linear time because it returns the value returned by the str method of either array deque or LL deque, which both run in linear time, as the number of steps increases as the input increases due to the presence of a loop.

No exceptions raised:

I would say there are benefits and drawbacks of not raising exceptions. I would say the main benefit is that it forces us to really think about the stress cases, as that is when most exceptions are raised. This allows the program to continue running even if there has been an error made and continue working as intended. This is a massive benefit, as this means that for the 'crash' cases for which we have wrote specific code to handle our program won't crash. A disadvantage would be that one, our program became more difficult to implement as solving these crash cases is not always trivial. Another disadvantage is that it can be more difficult to fix code, as error messages are specific and are useful for debugging. Overall, the benefit of not crashing as often seems to outweigh the disadvantages I listed above.

Test Cases:

The test cases which test stack and queues are in lines 346-561 and 589-800 respectively. I thought the most important things to test were the basic functions of both stacks and queues, and the stress cases. The basic functions are easy enough to check. Testing the basic cases includes, push/enqueue, pop/dequeue, and peeking one or multiple values. The more difficult cases to test are the stress cases. These special cases include things like growing, removing until things are empty, and making sure the attributes of the stack or deque are correct when it becomes empty. In my test cases I strived to ensure my stacks and queues worked as intended by extensively testing the stress cases. For instance, testing growing multiple times, emptying an array after growing, and ensuring the attributes of an empty stack or queue are what they need to be. In this way I think my tests are complete as they not only extensively test the stress cases, but through the testing of the base cases and stress cases, the basic functions of the deque are tested thoroughly.