For each of your 5 tests, once you have found a sequence that produces an error, describe why that implementation is incorrect. Here, a simple description of the types of sequences which cause an error is sufficient. Additionally, propose some ideas about what might be wrong with the implementation. These ideas can be high level, it is very difficult to isolate the exact problem in black box testing.

For the following five tests, all implementations are assumed to be a linked list implementation. This is due to the fact that whether the implementation is an array or a linked list is impossible to determine without enqueuing and dequeuing a large amount of strings.

For the first test, I simply enqueued and dequeue a string in order to pass the test. This tells me that either the enqueue failed to add the string to the queue or the dequeue failed to retrieve the string from the queue. An example of an incorrect implementation is that the method failed to create a Node in the queue when the enqueue method was called, thus when dequeue was called there was nothing to dequeue. Another example would be that the dequeue simply returns null no matter the values in the queue.

For the second test, I enqueued two strings and dequeued expecting the first string I put in. I believe that the implementation for the second test is actually a stack because I tested enqueuing and dequeuing a single string first and that failed the test. From this I can tell that the enqueue and dequeue method have no errors in adding and removing from the data structure. However, when I tested enqueuing two strings and dequeuing the second string, that also failed the test. Thus, from this I can tell that the second test has a stack implementation and not a queue implementation.

For the third test, I enqueued a string then dequeued it and dequeued again expecting the null value. Afterwards, I enqueued and dequeued a different string subsequently causing the test to pass. I believe that the error in the implementation is when dequeuing the pointer is set incorrectly when it empties the queue. This is because I enqueue and dequeuing added and removed from the queue perfectly. However, once the same steps were repeated after the queue was emptied, the test passed.

For the fourth test, I enqueued a string then dequeued it and dequeued again expecting the null value. Similar to the third test I enqueued and dequeued a different string, thereby causing the test to pass. Because both the third and fourth test passed in the same manner, I believe that the same error in implementation occurred again. The error is that the pointer is set incorrectly after the queue is emptied.

For the fifth test, I enqueued two different strings and dequeued expecting both strings in the order they were entered. I believe that the reason why the implementation failed was because it fails to save more than one unique node in the queue. This could be an error in either the enqueue or dequeue method. When I dequeue expecting the first string it fails the test, but when dequeuing expecting the second string it passes the test. This could be caused by the dequeue method returning the first sting every time.