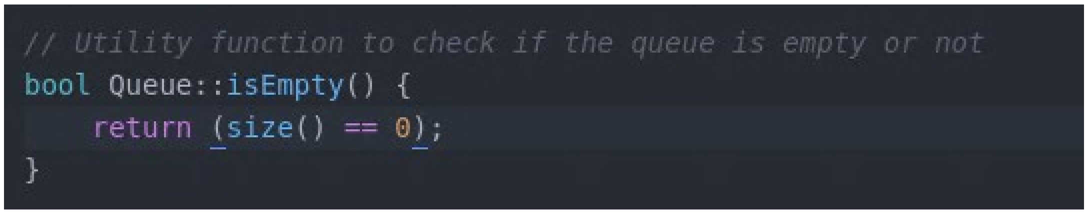
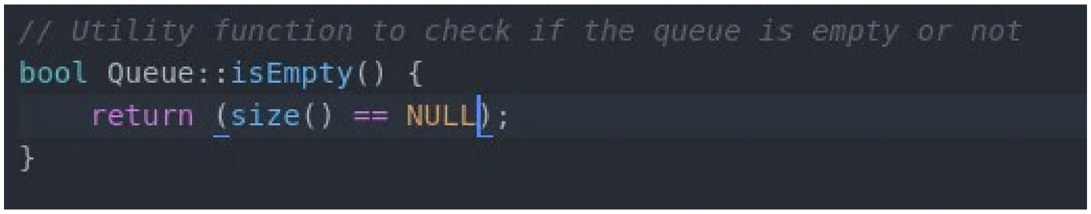
Tyler Johnson-Vasquez
joh20191@umn.edu
Section #04

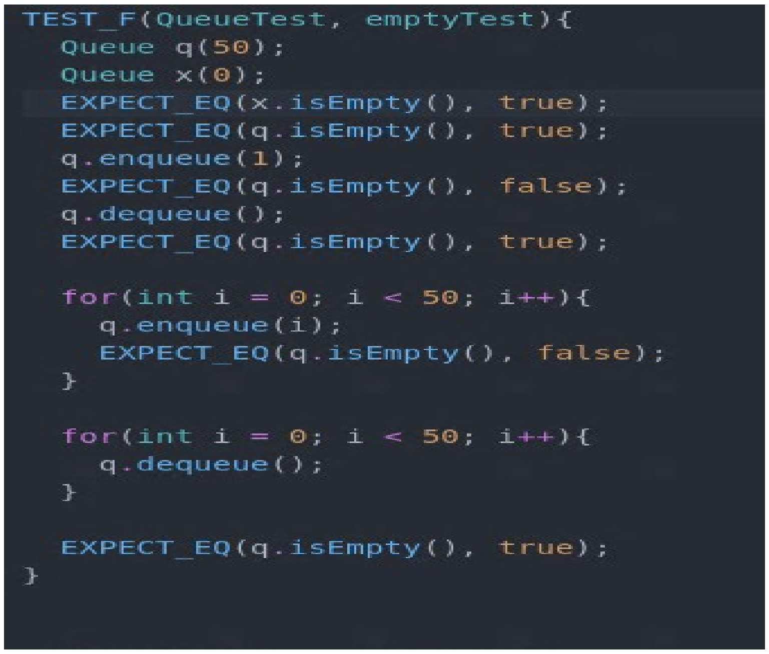
The mutant variant that I did was simply swap out a 0 with a NULL with the code snippets being below. The first snippet is the original and the second snippet is the edited version. I think it is important to test whether it is NULL or not because depending on what you are doing, it could lead to warning issues. One example is that if you are doing arithmetic and a person decides to use NULL as a zero, one would simply get a warning about it and could possibly run into overflow issues. NULL is supposed to be used as a NULL pointer constraint and not necessarily, although you can, as a substitute as a zero. I feel like having a test like this is important to get rid of pesky errors.

Although, the test suites that were given were very good at finding matter in the queue file. Even though the test suites did not pick up on the NULL issue, since NULL means that there is no value, the test suites would see through the NULL and just see it as a 0.


```
// Utility function to check if the queue is empty or not
bool Queue::isEmpty() {
    return (size() == 0);
}
```


```
// Utility function to check if the queue is empty or not
bool Queue::isEmpty() {
    return (size() == NULL);
}
```

In the queue_unittest1.cc, the empty test was conducted very well. Having two queues, testing if they are empty first then adding 1 to the q.queue and testing to see if it is empty again then taking it out of the queue and testing it again. Then using a loop up to 50 and testing it again then taking out the numbers that were just added and then testing it for the last time. This function was well written.


```
TEST_F(QueueTest, emptyTest){
  Queue q(50);
  Queue x(0);
  EXPECT_EQ(x.isEmpty(), true);
  EXPECT_EQ(q.isEmpty(), true);
  q.enqueue(1);
  EXPECT_EQ(q.isEmpty(), false);
  q.dequeue();
  EXPECT_EQ(q.isEmpty(), true);

  for(int i = 0; i < 50; i++){
    q.enqueue(i);
    EXPECT_EQ(q.isEmpty(), false);
  }

  for(int i = 0; i < 50; i++){
    q.dequeue();
  }

  EXPECT_EQ(q.isEmpty(), true);
}
```