**Name: Martin L. Metke**

**Problem Number:** 1.C

Execution State:

The combination of the “full” Semaphore and the “mutex” ensure that items can only be removed from the BoundedBuffer once one or more items has been inserted. Switching the order of lines 62 and 63 is not ideal, but still allows threads to safely insert and remove items.

Problem encountered:

This change should be benign – that is, it will not compromise concurrent execution of multiple threads – because while the “full” Semaphore will notify threads that there are items to be read, the mutex still will not allow them to enter the remove() method fully until the writer leaves the insert() method.

|  |  |
| --- | --- |
| Process #0 – insert() thread | |
| Line # | 63 ( was mutex.release(); ) |
| Relevant Variables | |
| mutex | After completing an insertion, the executing thread releases the mutex, signaling to another waiting thread that it may begin execution in some critical code. |

|  |  |
| --- | --- |
| Process #1 – remove() thread | |
| Line # | 63 ( was mutex.release(); ) |
| Relevant Variables | |
| mutex | Given a thread calling remove() immediately after the insert() thread released the “full” Semaphore, there is still no concurrency issue because execution does not begin until the insert() thread releases its mutex. |