**Name: Martin L. Metke**

**Problem Number:** 1.B

Execution State:

The “mutex” variable ensures that only one thread can enter the BoundedBuffer code sections which change the internal state of the buffer.

Problem encountered:

If the state-changing code sections were not protected by both the empty/”full” Semaphors and the “mutex”, then it would be possible for multiple threads to enter the insert() or remove() code blocks simultaneously, leading to concurrency issues where the state of the BoundedBuffer is incorrect (counts are off, inserted items are removed, internal pointers are off.)

|  |  |
| --- | --- |
| Process #0 – first insert() thread | |
| Line # | 50 ( was mutex.acquire(); ) |
| Relevant Variables | |
| Mutex | Previously, this variable would have allowed only one thread to execute within the insert() method at a time. |

|  |  |
| --- | --- |
| Process #1 – second insert() thread | |
| Line # | 50 ( was mutex.acquire(); ) |
| Relevant Variables | |
| Mutex | Now there is nothing preventing two or more threads from manipulating the internal variables of the BoundedBuffer, perhaps out of order, leading to (among other issues): - out-of-order entry insertion - overwriting of existing buffer entries  - overwriting of buffer entries inserted simultaneously |