**Programming Assignment 4 Report**

**Names: Andrew Ortiz and Mitchell Curtis**

# Problem Description

In this programming assignment, our goal is to implement a solution to the bounded-buffer problem using the producer-consumer process via mutex locks. We are instructed to use standard counting semaphores for empty, full, and a mutex lock. We’re instructed to manipulate this buffer via functions insert\_item() and remove\_item().

# Program Design

Our program consists of a buffer class that contains the previously mentioned insert\_item() and remove\_item() functions, as well as the proper getter/setter/print functions. The insert\_item() function bubble ups all the items already in the buffer, the remove\_item() does the same but with bubble down.

A screen shot of a computer program

Description automatically generated

# System Implementation

In the main.cpp we have a producer() function and a consumer() function that each get their own active thread in the main() function to randomly produce threads to ensure our solution is handling the bounded-buffer problem correctly.

A screen shot of a computer program

Description automatically generated

The threads used for both of these loops are created in the producer() and consumer() functions. These also create the items to be inserted and assign a random wait time for each thread created.

A screen shot of a computer program

Description automatically generated

**Results**

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

# Conclusion

The bounded-buffer problem is a complex issue that must be solved in order for operating systems to share resources effectively and efficiently. The consumer-producer process combined with mutex locks are a good way of solving this issue as the threads get halted when they’re a possibility of a race condition. Synchronization methods like this are essential for modern operating systems to work correctly.