CS5323 – Operating Systems II
Programming Assignment 1
Due: February 22, 2023, 11:59 p.m.
Submission: via Canvas

In this assignment, you will create a process that uses threads to implement a multiprogramming scenario and provide synchronization between these threads to ensure data consistency. You will create four threads – 2 producers and 2 consumers, who will have access to a shared circular buffer of size 20. Each item in the buffer should hold 1 integer. You can use an integer array as your shared buffer. Each of the producer threads must update a slot in the buffer with values from 1 to 30 (using a for loop with 30 iterations). Each of the consumer threads must update a shared variable called SUM by reading the buffer one item at a time. Each consumer can read any item in the buffer, but only 1 at a time. Each item should be read by exactly 1 consumer. Naturally, each consumer should read the shared buffer only when at least one slot is full and each producer should write to the shared buffer when at least one slot is empty.

In summary, you program must have a shared buffer and the shared variable called SUM. You can create other shared variables as needed. You can use either semaphores or mutexes to ensure that the threads are synchronized with each other. After all the producers and consumers have completed the execution, you should print the value of the shared variable SUM. The value of SUM, if your program is void of race conditions, should be 930.

Deliverables:

1. **Well-documented, readable code written in C. Clearly mark the entry section, critical section, exit section and remainder section in your code. You code will be executed on CSX so ensure that it works on there.**
2. **A README file with instructions on how to run your code.**
3. **A short report (~1 page) on how the synchronization logic works in your implementation and a justification on your choice between mutexes and semaphores.**