CSC415 OPERATING SYSTEM PRINCIPLES

Assignment 5
Producer-Consumer Problem

For this assignment, you will implement a multi-threaded producer-consumer program using a bounded buffer. The number of producers, the number of consumers, and the number of items each producer produces should be specified as command-line parameters. Thus, the following command should generate 2 producers, 4 consumers, and 32 items produced by each producer (assuming that the executable file is "ProConCode.out"):

```
./ProConCode.out 2 4 32
```

The buffer has 16 slots in total. Use full and empty semaphores to keep track of the numbers of full and empty slots in the buffer, respectively. Use a mutex to coordinate access of producers and consumers to the buffer. The main thread should parse all of the command-line parameters, initialize the synchronization objects, spawn all of the threads, and wait for them to complete. The consumers need to each consume the same number of items before exiting. Your code should calculate the number of items each consumer needs to consume so that every item produced gets consumed. (Your program does not need deal with the edge case where the consumers cannot equally share all the items.) The items produced by the producer threads are integers, obtained using the following expression:

```
thread_index * num_produced + counter
```

where `thread_index` is an integer passed to each producer thread as its index (we index all the producers from 0 to the number of producers minus one. `num_produced` is the number of items produced by each producer (i.e., 32 in the example), and `counter` is a local variable in each producer thread that gets initialized to 0 and incremented every time a new item is produced. For example, when each producer produces 32 items, the first item produced by the first producer with `thread_index = 0` is (0 * 32 + 0) = 0, the second item produced by this producer is (0 * 32 + 1) = 1, …, and the last item produced by this producer is (0 * 32 + 31) = 31.
The consumer threads should consume these items by simply printing them. (Printing is the only required "consumption".)

Use the functions `pthread_mutex_init()`, `pthread_mutex_lock()`, `pthread_mutex_unlock()`, `sem_init()`, `sem_wait()` and `sem_post()` for synchronization in POSIX.

Name your C file as “ProConCode”. Submit the source code file to the regular submission link on iLearn.