CPSC 457 - Winter 2017
Assignment 5
Due: April 12th @ 11:59 PM
Total: 20 marks; Worth 8% of your final mark

Objective:
To work with concurrency using pthreads.

Write a multithreaded producer-consumer program using the POSIX interface, with 10 producer threads and 1 consumer thread. Each producer thread sends 10 messages to the consumer before exiting. The consumer thread prints the contents of each received message on a separate line. The consumer thread exits after receiving 100 messages, 10 messages from each of the 10 threads. Each producer has a unique ID from 1 to 10, which are passed as arguments to the producer threads. The message that is sent by each producer is simply the producer ID.

The producers and the consumer threads share a "bounded blocking multiple-producer-single-consumer" queue. The queue is defined as follows:

```
#define MAX_QUEUE_SIZE 20
typedef struct queue{
      int element[MAX_QUEUE_SIZE];
      uint8_t  head;
      uint8_t  tail;
      uint8_t  remaining_elements; // # of elements in the queue
      // any more variables that you need can be added
} prod_cons_queue;

void queue_initialize( prod_cons_queue *q );
// initialize all queue variables and set element pointers to
// NULL
void queue_add(  prod_cons_queue *q,  int element );
int queue_remove(  prod_cons_queue *q );
// the removed element is returned in a double pointer "data"
```

You will write a thread-safe implementation of the "bounded blocking multiple producer single consumer" queue using pthread_mutex.

1. The **pthread_mutex_lock()** and **pthread_mutex_unlock()** calls are provided by the POSIX library to protect the queue operations of add() and remove().

2. Any producer thread that finds the queue full is blocked until there is a free entry in the queue. The consumer thread is blocked if the queue is empty. Blocking requires a thread to be suspended freeing the CPU. A blocked producer thread is unblocked when the consumer removes an item. The blocked consumer thread is unblocked when the producer thread adds an

item to the queue. To unblock and block threads use the **pthread_cond_signal()** and **pthread_cond_wait()** calls, respectively.

3. Your implementation must ensure that a blocked producer thread is given priority over a newly arriving thread. For instance, if a consumer remove() results in freeing a location in the queue, while there are waiting producer threads. The waiting producer threads must be able to add() items to the queue before a newly arriving producer thread.

4. Do not use any global variable in your program

Marking scheme:
Queue Implementation: 10
Implementation of Producer and consumer: 5 (2.5 each)
Implementation of main() function: 3
Output: 2

Deliverables:
Upload your code to d2l.
Provide hardcopy of your code and output. The code MUST BE arranged in the following order and sectioned: Implementation of the queue, implementation of producer and consumer threads, main function and finally the output. Hard copy submissions must have a cover page and a table of contents page mentioning page numbers for each section as mentioned.

In the hardcopy, please mention only your student ID.

Administrative Information

Teams

You are advised, but not required, to work in a team of two members. Teams of more than 2 students are not allowed. You and your partner must be in tutorials that are taught by the same TA. One submission per team is required.

Academic misconduct

Any similarities between assignments will be further investigated for academic misconduct. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your own original work. Any re-used code of excess of 10 lines must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

Late submission policy

Late submissions are penalized with 12.5% deduction for each late day or portion of a day. Hence no submission are accepted 8 days after the deadline

D2L Marks

Any marks posted on D2L or made available using any other mean are tentative are subject to change (after posting). They can go UP or DOWN due to necessary corrections.

Happy Coding :)