# BCIT

**Comp 4956 Software Engineering**

**Technical Programming Option**

# Option Head Mirela Gutica

**Fall 2022**

Mark: \_\_\_\_\_\_\_\_ /100

Lab 3 Concurrency Pattern: Producer Consumer

This is an **individual project**. The assignment should be handed-in based on the D2L deadline**. No late assignments will be accepted**.

This is a producer-consumer synchronization pattern with a circular buffer.

**Attacking the Problem:**

1. Read the PP presentation, the documentation/tutorials and the links that are included.
2. Observe and understand the console application class example **ProducerConsumerSemaphores** and **ProducerConsumerUsingMonitors.**
3. (25p) Explain the functionality of the two applications Produces/Consumer with semaphores and monitors.

**ProducerConsumerUsingMonitors Program**

This program is a basic producer and consumer as threads which only sleep and add/remove items from the buffer. The producer and consumer both access a shared circular buffer for the producer to add items (produce) and the consumer to remove items (consume). Monitor synchronization takes place within the program’s (synchronized) buffer when adding and removing items. Before entering the critical section of the program (adding or removing items in the buffer), the Monitor’s Enter method is used to acquire a lock (of the object) for the critical section so other threads will not be able to access this resource at the same time. The producer will check if the buffer is full, if it is, it will use the Monitor’s Wait method to release the lock so that the consumer thread may access this resource. If the buffer is not full, the producer will add an item (produce) to the buffer, and then uses the Pulse then Exit methods of the Monitor class. The Pulse method signals to waiting threads that the locked state of the object has changed, and the lock will be released. The waiting threads are put into a queue because of the method and will essentially wait their turn to receive the buffer’s lock object. Once obtained, the next waiting thread may check the new state of the buffer. The Exit method simply releases the lock of the buffer and marks the end of the critical section.

**ProducerConsumerSemaphores Program**

In this program, the teacher is the producer producing assignments and the consumer is the student who works (consumes) on the assignment until completion. Semaphore objects are initialized for when the resource pool is full and empty, then teacher and student threads are started. The teacher always makes assignments. Before producing, the teacher thread is blocked by the WaitOne() method of the Semaphore class, which checks its wait criteria is met. If it is not met, the thread will halt its execution until it is, and once it is met, the teacher will then sleep a random time to simulate time to produce an assignment. The number of the assignment is then put into a circular buffer to be consumed by the student thread. The teacher thread then releases the lock (Release() method) on the empty semaphore and then checks to see if the number of assignments will be reset. The student thread also continually consumes but it waits (calls semEmpty.WaitOne()) on the empty semaphore to not be empty. The student then generates a random amount of sleep time to simulate working on the assignment. Then it releases the semFull semaphore.

1. (75p) Build a multithreaded process (application) that has two threads: a thread that is used by an instructor to post assignments in a buffer (finite circular buffer with a fixed number of slots) and a thread that is used by a student to take and solve them.
2. You have to create a **GUI application in .NET and C#.**
   1. The assignment should be solved using the synchronization with semaphores pattern.
   2. The buffer is originally empty. If there are no empty slots where to place a new assignment, then the instructor has to wait. If there are no assignments, then the student has to wait.
   3. If a thread reaches the end, it will go at the beginning of the buffer (circular).
   4. When the student takes an assignment the slot will become free, therefore a new assignment can be later posted by the teacher.
   5. Use the Sleep function with a random number of time unis for both threads to simulate the production or consumption.
   6. You need to use the Threading name space.
   7. To create the buffer, use the C# Queue class.
   8. The buffer needs 2 methods: Put and Get. Make sure that you use a locking mechanism to prevent race condition.
   9. The Console display is thread safe; however, there is an issue in C# of accessing GUI components (i.e. text boxes) from multiple threads. The issue is related to the fact that Windows forms are not thread-safe: a control that is modified from multiple threads is not guaranteed to be modified correctly. In order to access controls correctly, one solution is to use delegates. Use the Invoke method of a control with a delegate. See examples at:

<http://msdn.microsoft.com/en-us/library/ms171728.aspx>