ETEC3702 – Concurrency
Lab 3 – Semaphores

Due date: 4 February 2020 by the end of class.

For this lab you are going to write a concurrent program that simulates a concurrent producer/consumer interaction.

Define functions as follows:

```
def producer():
   # This function will produce integers 1 through 100.
   # Each integer will be assigned, one at a time to a
   #   shared variable “buffer”.
   # After each number is produced, the program should sleep for between
   #   0.001 and 0.01 seconds.
   # Once all numbers have been produced, print “Producer done.”

def consumer():
   # This function will consume and print 100 values.
   # Each value will be retrieved by reading the
   #  shared variable “buffer”.
   # Print the number that was read.
   # After each number is printed, the program should sleep for between
   # 0.001 and 0.01 seconds.
   # Once 100 numbers have been consumed, print “Consumer done.”
```

Part 1: Concurrent Execution

- Write a program that creates threads for the producer and consumer threads, waits for them to complete, then prints “Done.”.
- Execute the program several times and observe the output.
- Are all 100 values displayed properly?
- Did each run produce the same results?
- Comment out the sleep lines in both functions and execute the program again.
- How did this change the output?

Part 2: Concurrent Execution with Semaphores

- Add the sleep instructions back in and add semaphores to the program so that the shared variable write / read operations are protected in each thread.
- Execute the program several times and observe the output.
- Are all 100 values now displayed properly?
- Did each run produce the same results?
- Comment out the sleep instructions again.
- How did this change the output?