**Programming Assignment 4**

**Thread Synchronization**

# Objective

The objective of this assignment is to use semaphores to protect the critical section between two competing threads.

# Assignment: Using Threads and Mutex/Counting Semaphores

The idea is to write a C/C++ program that creates two threads. The first thread is the consumer thread that consumes the data written to a shared memory buffer. The second thread is the producer thread that “produces” the data for the shared memory buffer. In order to prevent a race condition (e.g. the consumer reading before the producer writing) use a mutex semaphore and counting semaphores to coordinate when each thread can safely write or read to/from a common shared memory region.

**Prodcon Program Implementation**

The producer/consumer program (prodcon.c) that takes three arguments from the command line (no prompting the user from within the program).

1. To start the prodcon program

*./****prodcon*** *<memsize> <ntimes>* where *<memsize>* determines the overall size (multiple of 32) and total number of blocks (*memsize/32*) of the shared memory region. The argument *<ntimes>* indicates the number of times the producer writes to and the consumer reads from the shared memory region.

1. The main process is to create the shared memory region from the heap based upon *memsize*, initialize the mutex and counting semaphores and create both the producer and consumer threads then wait for both threads to finish.

1. The producer thread is to create *30 bytes* of random data (0-255) then store the checksum (use the Internet checksum) in the last 2 bytes of the shared memory block. The producer is to do this *ntimes* synchronizing with the consumer.

1. The consumer thread is to read the shared memory buffer of *30 bytes*, calculate the checksum based upon the *30 bytes* and compare that with the value stored in the shared memory buffer to ensure that the data did not get corrupted. The consumer is to do this *ntimes* synchronizing each read with the producer.

**Error Handling**

Perform the necessary error checking to ensure the correct number of command-line parameters. Limit *memsize* to 64K and ensure *ntimes* is a positive integer. If the consumer detects a mismatched checksum it is to report the error along with the expected checksum and the calculated checksum and exit the program.

# Grading

The program will be graded on the basic functionality, error handling and how well the implementation description was followed. Be sure to name your program ***prodcon.c*** (no extra characters, capitals) Note that documentation and style are worth 10% of the assignment's grade!

# Submission

The source code for program should be available on Blackboard.