# Semaphores, Mutex, and Conditions

## Part 1 – Thread Startup and Signal/Wait

In this part, create two threads: ping and pong. Ping sends a message (a simple integer) Pong. To send the message, Ping locks a mutex, increments a shared integer in memory, and unlocks the mutex. Ping then uses a *pthread* condition variable to signal to Pong that there is a new value available. Pong waits for the signal from Ping, and then prints the number.

There is only possible race condition: if Pong starts before Ping, it won’t be waiting for the first signal, and then the message will get lost. To fix this, we can create a semaphore, that is initially zero. When Ping starts up, it will try to sem\_wait, but won’t be able to pass the semaphore until Pong has started and is ready, and then Pong does a sem\_post to let Ping get started.

Sample Output:

PING: 1

PONG: 1

PING: 2

PONG: 2

PING: 3

PONG: 3

PING: 4

PONG: 4

PING: 5

PONG: 5

PING: 6

PONG: 6

PING: 7

PONG: 7

PING: 8

PONG: 8

PING: 9

PONG: 9

PING: 10

PONG: 10

For full credit you must pass the autograder, and you must have proper pthread mutex, condition variables, threads, and a semaphore. I need to see that you’ve used all three.