COMP 3413 Operating Systems Lab 3 Hand–in

Israelson	James	Mark: 20
Family Name	Given Name	
3632260		
Student Number		

[2] 1a. Initializing a pthread mutex variable: The first way to initialize a pthread mutex variable is to call pthread_mutex_init(mutex, attr). The mutex variable is initialized using the specified attributes. The second way to initialize a pthread mutex variable is to set mutex = PTHREAD_MUTEX_INITIALIZER. This is used to initialize mutexes that are statically allocated.

[2] 1b. `pthread_mutex_destroy()`: Yes. It can be reused. A destroyed mutex object can be reinitialized using pthread_mutex_init().

[3] 1c. `pthread_mutex_trylock()`: The function is equivalent to pthread_mutex_lock(), except if the mutex object is currently locked, the call returns immediately. It could be useful for checking if a mutex object is currently locked.

[3] 2a. `mutex_recursive.c`: The program prints "locking" two times and then hangs forever. I think it executes incorrectly because the thread is blocking after trying to lock an already locked mutex.

[3] 2b. modified `mutex_recursive.c`: The PTHREAD_MUTEX_RECURSIVE mutex can be used to successfully implement this program. The results of the newly edited program are to print "locking" 10 times diagonally to the right and then "unlocking" ten times diagonally to the left. The program then ends with "End of processing." This type of mutex would be helpful for a program that needs to access a mutex in a recursive function.

[1] 3a. expected value of shared variable: The shared variable should be equal to 20 x 50 = 1000

[1] 3b. results of 10 runs before changes: 136, 187, 188, 206, 176, 222, 253, 199, 197, 198. The variable is not being locked before it is accessed so there is a race condition causing an incorrect calculation.

[5] 3c. results of 10 runs after changes: All 10 runs the shared variable was equal to 1000. This is what I expected in 3a.