Santiago Bonada
Dan Hope
Rhys Agombar
Colton Howe

Tutorial 5
Conceptual Questions
====================
1.
pthread_create - creates a new pthread which will start execution within the function passed in as an argument of pthread_create.
pthread_join - the calling thread will wait until the thread specified in join completes execution.
pthread_exit - called within a thread, ends execution of the thread and returns the argument within to the thread which it will join with.

2.
Threads share memory that is within the process as a whole such as global variables or shared system resources, but threads have their own stack pointers, program counters, and registers.

3.
Differences between multithreading and multiprocessing:
-forking processes requires a lot of overhead
-child processes are clones of their parent process
-threads are very lightweight to create as they only duplicate required resources
-threads are killed when their parent process ends while child processes can continue execution.
-threads can more easily share resources with other threads

4.
A critical section is an area within the program where threads will change information and have the possibility of creating race conditions if other threads are manipulating the data at the same time.
Mutual exclusion is a form of controlling thread access to data by locking and unlocking access to threads which create request to use the resource and then signal that they have released the resource.

5.
pthread_mutex_t [name] - creates a mutex variable which manages resource locking
pthread_mutex_init - initializes the mutex variable
pthread_mutex_lock - the calling thread request for a lock on the mutex variable. If the mutex is currently locked then the calling thread has to wait until the mutex is unlocked.
pthread_mutex_unlock - the calling thread unlocks the mutex
pthread_mutex_trylock - same as pthread_mutex_lock but if the mutex is already locked then a "busy" error code will be returned.