In this lab we learned about concurrent programming using pthread. We learned that using create_pthread without join_pthread can cause problems as the main process may finish before the pthreads that were created. If you use join_pthread main will not complete until join_pthread has returned. We also learned about pthread_cond. This is a conditional pthread that can be used to advance a function or cause it to wait depending on a conditional variable. pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex) will cause a thread to wait until the condition specified is signaled and it unlocks the mutex which should have been locked at the beginning of the function to prevent concurrent access to the pthread. You can then use pthread_cond_signal(pthread_cond_t *cond) to send a signal to one thread that has that condition variable. If there is more then one thread there is no way to specify which the signal will be sent to using the command. Another way to do it is use pthread_cond_broadcast(pthread_cond_t *cond) which will unlock all threads that have the condition which can be useful as in the case of exercise three with the consumer.

3.1
In this program before anything is changed the thread2 function and main program print but the thread1 function does not.

I cannot see the threads output I only see the mains output. This is because the threads do not finish before the main program finishes which means they are terminated before they can complete.

The output is the printf from thread1, then thread2, and finally the main process. This is because the pthread_join function waits for the thread named to terminate before it returns. In this case it waited for thread1 and the second statement waited for thread2 which resulted in both of them being printed before the main process terminated.

3.2.1
When I compiled and ran t1.c the output of v was 0.

When I ran it a second time the value of v was -990. I believe this is because when you create the thread they both run and since the mutex is not locked it is a concurrent access. So instead of incrementing and then decrementing it does it at the same time. So when pthread_join is run for increment it makes v 990 but when it is run for decrement instead of changing the v value it replaces it.