```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define NUMTHREADS 3

//See if you successfully managed to gain the mutex lock
#define checkResults(string, val) {

        if (val) {
                printf("Failed with %d at %s", val, string);
                exit(1);
        }
}


pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int sharedData = 0;
int sharedData2 = 0;

//Thread function here
void *theThread(void *threadid){
        int rc;

        printf("Thread %.8x: Entered\n", (int)threadid);

        //Attempt to lock the mutex here
        rc = pthread_mutex_lock(&mutex);
        checkResults("pthread_mutex_lock()\n", rc);

        /********** Critical Section ******************/
        printf("Thread %.8x: Start critical section, holding lock\n",(int)threadid);

        /* Access to shared data goes here */
        sharedData++;
        sharedData2--;

        printf("Thread %.8x: End critical section, release lock\n",(int)threadid);
        /********** Critical Section ******************/

        //Release mutex
        rc = pthread_mutex_unlock(&mutex);
        checkResults("pthread_mutex_unlock()\n", rc);

        return NULL;
}

int main(int argc, char **argv){

        pthread_t thread[NUMTHREADS];
        int rc=0;
        int i;

        //Lock the shared mutex
        rc = pthread_mutex_lock(&mutex);
        checkResults("pthread_mutex_lock()\n", rc);

        //Create threads here
        for (i=0; i<NUMTHREADS; ++i) {
                rc = pthread_create(&thread[i], NULL, theThread, (void *)i);
                checkResults("pthread_create()\n", rc);
        }

        printf("Wait a bit until we are 'done' with the shared data\n");
        sleep(3);


        //Unlock the shared mutex
        rc = pthread_mutex_unlock(&mutex);

        //Wait for threads to finish and release their resources
        for (i=0; i <NUMTHREADS; ++i) {
                rc = pthread_join(thread[i], NULL);
                checkResults("pthread_join()\n", rc);
```

```
        }

        printf("Results: sharedData: %d, sharedData2: %d\n",sharedData,sharedData2);

        //Free up memory
        rc = pthread_mutex_destroy(&mutex);
        return 0;
}
```