```c
/**
 * mcpi.c
 *
 * CS 470 Pthreads Lab
 * Based on IPP Programming Assignment 4.2
 *
 * Names: Onyx, Josh Derrow, Brennan Krutis
 */

#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#include "timer.h"

int thread_count;               // thread count
long total_darts = 0;           // dart count
long darts_in_circle = 0;       // number of hits
pthread_mutex_t barrier_mutex;

void* throw_darts(void* arg)
{
    // seed pseudo-random number generator (TODO: use thread ID as seed)
    unsigned long seed = pthread_self(); // changed for Q3

    long local_darts = total_darts/thread_count; // added for Q3
    long local_darts_in_circle = 0; // added for Q8
    for (long dart = 0; dart < local_darts; dart++) {

        // throw a dart by generating a random (x,y) coordinate pair
        // using a basic linear congruential generator (LCG) algorithm
        // (see https://en.wikipedia.org/wiki/Linear_congruential_generator)
        //
        seed = (1103515245*seed + 12345) % (1<<31);
        double x = (double)seed / (double)ULONG_MAX;
        seed = (1103515245*seed + 12345) % (1<<31);
        double y = (double)seed / (double)ULONG_MAX;
        double dist_sq = x*x + y*y;

        // barrier set up added for Q6
        //int counter = 0;

        // update hit tracker
        // changed for Q6
        // changed again for Q8
        if (dist_sq <= 1.0) {
            local_darts_in_circle++;
        }
    }
    // added for Q8 (less calls to global is less competition)
    pthread_mutex_lock(&barrier_mutex);
    darts_in_circle += local_darts_in_circle;
    pthread_mutex_unlock(&barrier_mutex);

    return NULL;
}

int main(int argc, char* argv[])
```

```c
{
    // check and parse command-line arguments
    if (argc != 3) {
        printf("Usage: %s <num-darts> <num-threads>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    total_darts = strtoll(argv[1], NULL, 10);
    thread_count = strtol(argv[2], NULL, 10);

    START_TIMER(darts)

    // init mutex (global variable) added for Q6
    pthread_mutex_init(&barrier_mutex, NULL);

    // simulate dart throws (TODO: spawn multiple threads)
    pthread_t thread_handles[thread_count]; // changed for Q3

    // changed for Q3
    for (int i = 0; i < thread_count; i++) {
        pthread_create(&thread_handles[i], NULL, throw_darts, (void*)0);
    }
    for (int i = 0; i < thread_count; i++) {
        pthread_join(thread_handles[i], NULL);
    }

    STOP_TIMER(darts)

    // calculate pi
    double pi_est = 4 * darts_in_circle / ((double)total_darts);
    printf("Estimated pi: %e   Time elapsed: %.3lfs  w/  %d thread(s)\n",
            pi_est, GET_TIMER(darts), thread_count);

    // clean up
    pthread_mutex_destroy(&barrier_mutex);
    return EXIT_SUCCESS;
}
```