```c
/* File:     many_mutexes.c

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h> /* for clock_gettime */
#include <stdint.h> /* for uint64 definition */

#define BILLION 1000000000L

int thread_count;
int n;
int total = 0;
pthread_mutex_t mutex;
uint64_t diff;

void Usage(char prog_name[]);
void* Lock_and_unlock(void* rank);

int main(int argc, char* argv[]) {
   pthread_t* thread_handles;
   long thread;
   struct timespec start, end;

   if (argc != 3) Usage(argv[0]);
   thread_count = strtol(argv[1], NULL, 10);
   n = strtol(argv[2], NULL, 10);

   thread_handles = malloc(thread_count*sizeof(pthread_t));
   pthread_mutex_init(&mutex, NULL);

    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start);  /* mark start time */

   for (thread = 0; thread < thread_count; thread++)
      pthread_create(&thread_handles[thread], NULL, Lock_and_unlock,
            (void*) thread);

   for (thread = 0; thread < thread_count; thread++)
      pthread_join(thread_handles[thread], NULL);

    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &end);    /* mark the end time */

   printf("Total number of times mutex was locked and unlocked: %d\n",
         total);

    diff = BILLION * (end.tv_sec - start.tv_sec) + end.tv_nsec -
  start.tv_nsec;
  printf("elapsed process CPU time = %llu nanoseconds\n", (long long
```

```
  unsigned int) diff);
60
61    pthread_mutex_destroy(&mutex);
62    free(thread_handles);
63    return 0;
64 }  /* main */
65
66 /*--------------------------------------------------------------------
67  * Function:   Usage
68  * Purpose:    Print a message explaining how to start the program.
69  *             Then quit.
70  * In arg:     prog_name:  name of program from command line
71  */
72 void Usage(char prog_name[]) {
73    fprintf(stderr, "usage: %s <thread_count> <n>\n", prog_name);
74    fprintf(stderr, "    n: number of times mutex is locked and ");
75    fprintf(stderr, "unlocked by each thread\n");
76    exit(0);
77 }  /* Usage */
78
79
80 /*--------------------------------------------------------------------
81  * Function:   Lock_and_unlock
82  * Purpose:    Repeatedly lock and unlock a mutex to determine performance
83  *             of mutexes
84  * In arg:     rank:  thread rank
85  * In globals: thread_count:  number of threads
86  *             n:  number of times each thread should lock and unlock mutex
87  *             mutex:
88  * In/out global:  total:  total number of times mutex is locked and
    unlocked.
89  */
90 void* Lock_and_unlock(void* rank) {
91    long my_rank = (long) rank;  /* unused */
92    int i;
93
94    for (i = 0; i < n; i++) {
95       pthread_mutex_lock(&mutex);
96       total++;
97       pthread_mutex_unlock(&mutex);
98    }
99
100   return NULL;
101 }  /* Lock_and_unlock */
102
```