```c
/* File:     many_sems.c

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

#define BILLION 1000000000L

int thread_count;
int n;
int total = 0;
sem_t sem;
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));
   sem_init(&sem, 0, 1);

   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 sem 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);
 62
 63    sem_destroy(&sem);
 64    free(thread_handles);
 65    return 0;
 66 }  /* main */
 67
 68 /*---------------------------------------------------------------------
 69  * Function:   Usage
 70  * Purpose:    Print a message explaining how to start the program.
 71  *             Then quit.
 72  * In arg:     prog_name:  name of program from command line
 73  */
 74 void Usage(char prog_name[]) {
 75    fprintf(stderr, "usage: %s <thread_count> <n>\n", prog_name);
 76    fprintf(stderr, "    n: number of times semaphore is locked and ");
 77    fprintf(stderr, "unlocked by each thread\n");
 78    exit(0);
 79 }  /* Usage */
 80
 81
 82 /*---------------------------------------------------------------------
 83  * Function:   Lock_and_unlock
 84  * Purpose:    Repeatedly lock and unlock a semaphore to determine
   performance
 85  *             of semaphores
 86  * In arg:     rank:  thread rank
 87  * In globals: thread_count:  number of threads
 88  *             n:  number of times each thread should lock and unlock
   semaphore
 89  *             sem: the semaphore
 90  * In/out global:  total:  total number of times sem is locked and unlocked.
 91  */
 92 void* Lock_and_unlock(void* rank) {
 93    long my_rank = (long) rank;  /* unused */
 94    int i;
 95
 96    for (i = 0; i < n; i++) {
 97       sem_wait(&sem);
 98       total++;
 99       sem_post(&sem);
100    }
101
102    return NULL;
103 }  /* Lock_and_unlock */
104
```