```
/*
Name: John Robertson
BlazerId: jprob
Lab #: 12
To compile: gcc lab12.c -pthread
To run: ./a.out <int> <int>
*/
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;

typedef struct Penult {
    pthread_t ptid;
    int tid;
    int N, size;
    double* a;
    double sum;
} Penult;



void *compute(void *arg) {
    Penult* info = (Penult*) arg;
    int myStart, myEnd, myN, i;

    // determine start and end of computation for the current thread
    myN = info->N/info->size;
    myStart = info->tid * myN;
    myEnd = myStart + myN;
    if (info->tid == (info->size - 1)) {
        myEnd = info->N;
    }

    // compute partial sum
    double mysum = 0.0;
    for (i = myStart; i < myEnd; i++) {
        mysum += info->a[i];
    }

    // grab the lock, update global sum, and release lock
    pthread_mutex_lock(&mutex);
    info->sum += mysum;
    pthread_mutex_unlock(&mutex);

    return (NULL);
}

int main(int argc, char **argv) {
    long i;
    int N, size;
    double* a;
    Penult* info;

    if (argc != 3) {
        printf("Usage: %s <# of elements> <# of threads>\n", argv[0]);
        exit(-1);
    }

    N = atoi(argv[1]); // no. of elements
    size = atoi(argv[2]); // no. of threads
```

```
    // allocate vector and initialize
    info = (Penult*) malloc(sizeof(Penult) * size);

    a = (double*) malloc(sizeof(double) * N);
    for (i = 0; i < N; i++) {
        a[i] = (double) (i + 1);
    }

    // create threads
    for (i = 0; i < size; i++) {
        info[i].tid = i;
        info[i].size = size;
        info[i].N = N;
        info[i].a = a;
        pthread_create(&info[i].ptid, NULL, compute, (void *)&info[i]);
    }

    // wait for them to complete
    for (i = 0; i < size; i++) {
        pthread_join(info[i].ptid, NULL);
    }

    //sum the sums
    double master_sum;
    for (i = 0; i < size; i++) {
        master_sum += info[i].sum;
    }

    printf("The total is %g, it should be equal to %g\n", master_sum, ((double) N * (N+
1))/2);
    free(info);
    free(a);

    return 0;
}
```