```
/*
  To Compile: gcc -O -Wall jlplayerlab12.c -lpthread
  To Run: ./a.out 1000 4
*/

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

pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;

struct threadVari{

  double *a; // Points to the array
  double *sum; // Points to final sum
  int N; // The total number of elements
  int size; // The number of threads
  long tid; // The thread Identification Number
};

double *a=NULL, sum=0.0;
int N, size;


void *compute(void *arg) {
    struct threadVari *data = (struct threadVari *)arg;

    long tid  = data->tid;
    int N     = data->N;
    int size  = data->size;
    double *a = data->a;

    int myN = N / size;
    int myStart = tid *myN;
    int myEnd = myStart + myN;
    int i;


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

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

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

    return (NULL);
}

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

    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
```

```
    pthread_t *threads = malloc(size * sizeof(pthread_t));
    struct threadVari *params = malloc(size * sizeof(struct threadVari));

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


    // create threads
    for ( int i = 0; i < size; i++){
        params[i].a = a;
        params[i].sum = &sum;
        params[i].N = N;
        params[i].size = size;
        params[i].tid = i;

      pthread_create(&threads[i], NULL, compute, &params[i]);
    }


    // wait for them to complete
    for ( int i = 0; i < size; i++)
      pthread_join(threads[i], NULL);



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

    return 0;
}
```