```
/*
  Simple Pthread Program to find the sum of a vector.
  Uses mutex locks to update the global sum.
  Author: Purushotham Bangalore
  Date: Jan 25, 2009

  To Compile: gcc -O -Wall pthread_psum.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 Info {
  int tid;
  double *a;
  double *global_sum;
  int N;
  int size;
};



void *compute(void *arg) {
    //Convert back to struct
    struct Info *data = (struct Info *)arg;

    int myStart, myEnd, myN, i;
    //Initialize the variables to what's in the data struct
    long tid = data->tid;
    int N = data->N;
    int size = data->size;

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

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

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

    return (NULL);
}

int main(int argc, char **argv) {
    long i;
    pthread_t *tid;
    struct Info *thread_data; //Array of structs holding the info for each thread
    double *a=NULL, sum=0.0;
    int    N, size;

    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
    tid = (pthread_t *)malloc(sizeof(pthread_t)*size);
    thread_data = (struct Info *)malloc(sizeof(struct Info) * size); //Allocate a struct fo
r each thread (total num of threads = 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++) {
      //For each thread, fill its struct with the the data for this specific thread
      thread_data[i].tid = i;
      thread_data[i].a = a;
      thread_data[i].global_sum = &sum;
      thread_data[i].N = N;
      thread_data[i].size = size;
      pthread_create(&tid[i], NULL, compute, (void *)&thread_data[i]); //Convert the struct
 pointer to a void pointer and pass it
    }

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

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

    free(tid);
    free(a);
    free(thread_data);

    return 0;
}
```