```
/*
  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>


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

struct args {
    double *a;
    double sum;
    int N;
    int size;
    long tid;
};

void *compute(void *arg) {
    // Used the following to get the struct elements
    // https://www.tutorialspoint.com/cprogramming/c_structures.htm
    int myStart, myEnd, myN, i;
    long tid = ((struct args*)arg)->tid;
    double *a = ((struct args*)arg)->a;
    int N = ((struct args*)arg)->N;
    int size = ((struct args*)arg)->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++){
      // printf("My thread ID is %d, and mysum is %f\n",tid,mysum);
      mysum += a[i];

    }


    // grab the lock, update global sum, and release lock
    ((struct args*)arg)->sum += mysum;

    printf("My thread ID is %d, and my sum is %f\n",tid,((struct args*)arg)->sum);

    return (NULL);
}

int main(int argc, char **argv) {
    long i;
    pthread_t *tid;
    double *a;
    double sum = 0;
    int N;
    int 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

    struct args allarg[N];

    // allocate vector and initialize
    tid = (pthread_t *)malloc(sizeof(pthread_t)*size);
    a = (double *)malloc(sizeof(double)*N);
    for (i=0; i<N; i++)
      a[i] = (double)(i + 1);

    for (i=0; i<N; i++){
      allarg[i].N = N;
      allarg[i].sum=0;
      allarg[i].size = size;
      allarg[i].tid = i;
      allarg[i].a = a;
    }

    // create threads
    for ( i = 0; i < size; i++){
      pthread_create(&tid[i], NULL, compute, &allarg[i]);
    }


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

    for (i=0;i<size;i++){
      sum += allarg[i].sum;
    }

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

    return 0;
}
```