#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define MAXTHREADS 10

volatile unsigned long long Round = 0ULL;

volatile unsigned long long Total = 0ULL;

pthread\_mutex\_t sum\_mutex1 = PTHREAD\_MUTEX\_INITIALIZER;

void \*Summation(void \*tidptr)

{

unsigned long long local\_total = 0;

unsigned long long tid = \* ((unsigned long long\*) tidptr);

for (unsigned long long dex = 0ULL; dex <= tid; dex++) {

local\_total += dex;

}

pthread\_mutex\_lock(&sum\_mutex1);

Total += local\_total;

pthread\_mutex\_unlock(&sum\_mutex1);

return NULL;

}

void doOneRound(unsigned long long thisRound, int numthreads)

{

int tids[MAXTHREADS] = {0,1,2,3,4,5,6,7,8,9};

pthread\_t t[numthreads];

int rc;

/\* start all of the threads \*/

for (int i = 0; i < numthreads; i++) {

rc = pthread\_create(&t[i], NULL, Summation, (void \*) &thisRound);

if (rc){

printf("ERROR; return code from pthread\_create() is %d\n", rc);

exit(-1);

}

}

/\* wait for threads to complete \*/

for (int i = 0; i < numthreads; i++) {

rc = pthread\_join(t[i], NULL);

if (rc != 0) {

fprintf(stderr, "ERROR joining with thread %d (error==%d)\n", tids[i], rc);

exit(-1);

}

}

}

void checkResult(unsigned long long thisRound, int numthreads)

{

/\*

\* note: credit for this closed-form solution goes to Johann Carl Friedrich Gauss.

\* it will not work for very large values of thisRound

\* because the multiplication will cause it to overflow. but it is good enough

\* for our purposes.

\*/

unsigned long long calc = (thisRound \* (thisRound + 1ULL)) / 2ULL;

calc \*= (unsigned long long) numthreads;

if (Total != calc) {

printf("PARENT: ERROR! Round %llu total should have been %llu but was %llu\n", thisRound, calc, Total);

exit(-1);

}

}

int main(int argc, char \*argv[])

{

if (argc != 3) {

fprintf(stderr, "USAGE: %s <nthreads> <max>\n", argv[0]);

exit(-1);

}

int numthreads = atoi(argv[1]);

if ((numthreads < 1) || (numthreads > MAXTHREADS)) {

fprintf(stderr, "ERROR: numthreads must be >= 1 and <= %d\n", MAXTHREADS);

exit(-1);

}

unsigned long long numrounds = strtoull(argv[2], NULL, 0);

if (numrounds <= 0ULL) {

fprintf(stderr, "ERROR: number of rounds must be a positive unsigned long long (not '%s')\n", argv[2]);

exit(-1);

}

printf("PARENT: input %d threads %llu rounds\n", numthreads, numrounds);

for (Round = 1ULL; Round < numrounds; Round++) {

Total = 0ULL;

doOneRound(Round, numthreads);

checkResult(Round, numthreads);

}

printf("PARENT: SUCCESS! exiting after final Round %llu (Total: %llu)\n", numrounds-1, Total);

return(0);

}