#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <iostream>

using namespace std;

int num1 = 0;

int num2 = 0;

bool raceDetected;

unsigned int seed = 0x000fff;

const int NUM\_THREADS = 2;

pthread\_mutex\_t mutex;

void \* thread\_routine (void \*thread) {

int \*id\_ptr, thread\_num;

id\_ptr = (int \*) thread;

thread\_num = \*id\_ptr;

int counter = 0;

int tmp1;

int tmp2;

int r;

do {

tmp1 = num1;

tmp2 = num2;

r = rand\_r ( &seed ) % 11;

num1 = tmp1 + r;

num2 = tmp2 - r;

counter++;

} while ( num1 + num2 == 0 && !raceDetected );

raceDetected = true;

pthread\_mutex\_lock(&mutex);

cout << "counter: " << counter << " - thread number: " << thread\_num << endl;

pthread\_mutex\_unlock(&mutex);

}

int main() {

pthread\_t tid[NUM\_THREADS];

int\* threadIdNum[NUM\_THREADS];

for (int i=0; i<NUM\_THREADS; i++ ) {

threadIdNum[i] = new int;

\*threadIdNum[i] = i;

if ( pthread\_create ( &(tid[i]), NULL, thread\_routine, (void \*) threadIdNum[i] )) {

cout << "thread create failed!\n";

exit(1);

}

}

do {

0 + 0;

} while( !raceDetected );

return 0;

}