```c
/**
 * Homework 2: Parallel, binary max function
 *
 * CS4414 Operating Systems
 * Fall 2017
 *
 * Maurice Wong - mzw7af
 *
 * barrier.c - barrier implementation
 *
 */


#include "barrier.h"

/**
 * Takes a barrier b and initializes it to the correct size
 * Also initializes semaphores to correct values
 */
void initBarrier(barrier *b, int size) {
    b->size = size;
    b->counter = 0;
    sem_init(&b->waitq, 0, 0);
    sem_init(&b->waitq2, 0, 1);
    sem_init(&b->mutex, 0, 1);
}

/**
 * Causes a thread to wait on barrier b
 */
void waitBarrier(barrier *b) {
    sem_wait(&b->mutex);
    b->counter++;
    if (b->counter == b->size) {
        sem_wait(&b->waitq2); // lock barrier2 to force threads to wait for reset
         after release
        sem_post(&b->waitq);  // RELEASE THE THREADS!!!
    }
    sem_post(&b->mutex);

    // turnstile to release threads
    sem_wait(&b->waitq);
    sem_post(&b->waitq);
    // reset counter back to 0 so we can reuse the barrier
    sem_wait(&b->mutex);
    b->counter--;
    if (b->counter == 0) {
        sem_wait(&b->waitq); // lock the barrier again
        sem_post(&b->waitq2); // Done resetting , let threads continue
    }
    sem_post(&b->mutex);

    // turnstile to collect threads when resetting barrier:
    sem_wait(&b->waitq2);
```

```
    sem_post(&b->waitq2);
}
```