```
struct lock {
    int held = 0;
    queue Q;
}

void acquire (lock) {
     disable_interrupts();
     while (lock->held){
         enqueue(lock->Q, current_thread);
         thread_block(current_thread);
     }
     lock->held = 1;
     enable_interrupts();
}

void release (lock) {
    disable_interrupts();
    if (!isEmpty(lock->Q)){
        thread_unblock(dequeue(lock->Q));
    }
    lock->held = 0;
    enable_interrupts();
}
```

(good) implementation of a sleeping lock

Semaphore Implementation

```
struct semaphore {
    int value;
    queue Q;
}

void init(sema, value){
    sema->value = value;
}

void P (sema) {
    disable_interrupts();
    while (sema->value == 0){
        enqueue(sema->Q, current_thread);
        thread_block(current_thread);
    }
    sema->value--;
    enable_interrupts();
}

void V (sema) {
    disable_interrupts();
    if (!isEmpty(sema->Q)){
        thread_unblock(dequeue(sema->Q));
    }
    sema->value++;
    enable_interrupts();
```