1. [0.25 point] An instruction, e.g., `CompareAndSwap()`, that executes atomically
   
   a) must consists of only one machine instruction.
   b) executes as a single, uninterruptable unit.
   c) cannot be used to solve the critical section problem.

2. [0.5 point] What does spin waiting refer to?

   A thread waits in a loop to acquire a lock.

3. [2 points] Consider a stack that is implemented by an array. The following pseudocode shows `push()` and `pop()` operations for this stack. Assume that the stack is accessed by threads concurrently.

   a) Specify the name of a variable that has a race condition.
   b) How can the race condition be fixed?

```c
push(item) {
    if(top < SIZE) {
        stack[top] = item;
        top++;
    } else {
        ERROR
    }
}

pop() {
    if(!isempty()) {
        top--;
        return stack[top];
    } else {
        ERROR
    }
}

isempty() {
    if (top == 0) {
        return true;
    } else {
        return false;
    }
```
4. [2 points] We learned in the class that CompareAndSwap() instruction is used for implementing a spinlock:

```c
void lock(lock_t *lock) {
    while(CompareAndSwap(&lock->flag, 0, 1) == 1) // spin
}
```

A variation of implementing the spinlock is to check the availability of the lock before calling the CompareAndSwap() instruction:

```c
void lock(lock_t* lock) {
    while(true) {
        if(lock->flag == 0) { // lock is likely available.
            // A context switch occurs here.
            if(!CompareAndSwap(&lock->flag, 0, 1)) {
                break;
            } // break
        } // if
    } // while
}
```

Explain, with an example, how this variation provides mutual exclusion when a context switch occurs inside the lock() function between two threads, Thread 1 and Thread 2. Because of this context switch, assume that OS interrupts Thread 1 and runs Thread 2.

When Thread 2 runs, flag is 0. Then, Thread 2 sets flag to 1. When Thread 1 resumes execution, it finds that the flag is not 0. Thus, Thread 1 fails to acquire the lock and jumps back to the start of the while-loop.

5. [1 point] Explain how a single-CPU system is compromised by disabling interrupts as a way to implement locks.

    When the owner of a lock is not interrupted, other threads do not get opportunities to run.