ECSE 420 Assignment 2 Report
Group 19

Stuart Mashaal
260639962

Oliver Tse Sakkwun
260604362

Due: November 7, 2018

Question 1

1.2

Some stuff

1.4

Some stuff

1.5

Some stuff

Question 2

2.1 - LockOne

No. If the shared `flag` atomic registers are replaced by regular registers, `LockOne` does not still provide mutual exclusion. **Proof:**

```
                Thread 1                                   Thread 2

public void lock() {                        //
    int i = ThreadID.get();                 //
    int j = i - 1;                          //
    flag[i] = true // only local            //
//                                          public void lock() {
//                                              int i = ThreadID.get();
//                                              int j = i - 1;
//                                              flag[i] = true // only local
//                                              while (flag[j]) {} // false
//                                          } // enter critical section
    while (flag[j]) {} // false             //
} // enter critical section                 //
```

The problem is that if we don't use atomic registers, then the writes to `flag` are not necessarily shared right away. This means that both threads can write to `flag` without the other being able to detect it. Then, both can enter the critical section.

2.2 - LockTwo

No, if shared variable `victim` uses a regular register instead of an atomic one `LockTwo` does not still provide mutual exclusion, for similar reasons to those of 2.1. **Proof:**

```
                Thread 1                                   Thread 2

public void lock() {                          //
  int i = ThreadID.get();                     //
  victim = i //only local                     //
//                                            public void lock() {
//                                              int i = ThreadID.get();
//                                              victim = i //only local
//                                              while (victim == i){} //see t1 write
//                                            } //enter critical section
  while (victim == i){} //see t2 write        //
} // enter critical section                   //
```

Since the `victim` register is no longer atomic, writes to it by either thread are not only *not shared immediately* but also are *not sequentially consistent*. The Java Concurrency Model does not guarantee sequential consistency without use of synchronization primitives (such as the `volatile` keyword, which makes a register atomic).

Question 3

Some stuff