

Name _____

CMPSCI 377 Sample Midterm

This sample midterm is not to scale: there will be more short answer and true false questions.

1. Short Answer

a) What properties do locks and condition variables provide? What properties do semaphores provide? (use the terminology from class)

b) Explain what a wait queue is, how it interacts with the scheduler, and how it is used in locks and condition variables.

c) Can two threads running in separate threads corrupt one another's stack in C++? What about in Java? Explain.

Name _____

2. Synchronization

Compare and swap (CAS) is an operation that does the following:

```
1: bool CAS(x,y,z){
2:     if (x == z){
3:         tmp = y;
4:         y = x;
5:         x = y;
6:         return true;
7:     }
8:     else return false;
9: }
```

a) Implement lock and unlock using compare and swap (CAS). Points will be awarded for **correctness** and **simplicity**. Do not worry about performance.

b) Does compare and swap need to be atomic for this to work? If so, give an interleaving that breaks the code.

Name _____

3. Monitors

A popular synchronization method that we have not discussed in class is a **barrier**. The semantics of a barrier are the following: all threads who encounter a barrier must wait until all other threads reach that same barrier, then they can all continue.

I would like to modify this synchronization primitive a little to make barrier regions with two functions `barrier_enter` and `barrier_leave`. So the semantics of the barrier region are that if any thread reaches `barrier_leave`, it must wait until all other threads that have passed the `barrier_enter` reach the same barrier leave.

Implement these two functions with monitors. Do not worry about multiple barriers. Points for simplicity.

```
barrier_enter(){\n
```

```
\n}
```

Name _____

```
barrier_leave(){
```

```
}
```