CSE 410 - Computer Systems
Homework 3

Assigned:	Monday, November 26, 2001
Due:	Monday, December 3, 2001 At the start of class

Your name: ________________________________

1. On most modern systems, there are usually several "ready queues" and several "wait queues." None of the tasks (threads) on any of these queues is actually executing instructions. For the following questions, describe a set of circumstances that would cause the given scenario to take place. There are many such possibilities.

a. Describe a set of circumstances under which a task would move from a wait queue to a ready queue. Include a description of the particular wait queue before the transition, the event that triggers the transition, and a description of the particular ready queue after the transition.

b. Describe a set of circumstances under which a task would move from a ready queue to being the running task, executing instructions on a CPU. Include a description of a particular ready queue before the transition, the event that triggers the transition, and a description of how the scheduler picks this particular task to run next.

2. How does preemptive scheduling allow the operating system scheduler to have more control over system performance than non-preemptive scheduling?

3. For scheduling purposes, a useful characterization of tasks is often that they are "I/O bound" or "CPU bound".

a. Give an example of a desktop PC task that is likely to be I/O bound often. Why does this task fit this description?

b. Give an example of a desktop PC task that is likely to be CPU bound for relatively long periods. Why does this task fit this description?

4. The lecture dated November 21, 2001 is about scheduling on Windows 2000. On slide 8 there is a description of theWin2K priority structure, and on slide 11 there is a snapshot of perfmon4 and cpustres running on my laptop.

a. What is the process priority class for cpustres?

b. What is the most recent numeric value of the priority of cpustres thread 2?

c. What would the numeric value of the priority of cpustres thread 2 be if we changed the process priority class to Below Normal?

d. Looking at the snapshot, you can see that the priority of cpustres thread 1 jumps around quite a bit in the middle of the graph, before it finally settles down to a constant value of 6. What is the technical term that describes this jumping around? What is a likely explanation for why it happened at the point when it did?

5. Consider the slides in the November 21 lecture that describe various scheduling scenarios (slides 16 to 19). Note that the thread that comes off the ready queue to be run is always the one at the head of the queue. For this question, the term "scheduling events" means statements like "move to wait queue for device read", "quantum exhausted", "preempt by higher priority task" and so on.

a. Are the threads in these examples "dynamic" threads or "real time" threads?

b. Starting from the situation shown in slide 17, where thread B is running, describe a set of scheduling events under which thread C will get a chance to run.

c. Starting from the situation shown in slide 17, where thread B is running, describe a set of scheduling events under which thread E will get to run.

6. Coordinated access to shared state variables is the key issue in synchronization of multiple threads.

a. On a preemptively scheduled system, is the following **`if`** statement an "atomic operation", or is it possible for a thread to be interrupted while executing the statement?

```
if (j < k) j++;
```

b. Is the "**`j++`**" portion of the statement an "atomic operation" or is it possible for the thread to be interrupted while executing the statement?

c. Assuming that j and k are shared state variables, show how you would use a lock with this code to prevent corruption due to a context swap at an inconvenient moment?

7. Consider the lecture "Synchronization Part 2", dated November 28. On slide 12, there is an example of how Push() and Pop() functions can be implemented to provide coordinated, multi-thread access to a single shared stack.

a. Notice that the Push procedure uses **`condition->signal(lock)`** to alert any waiting thread that there is something on the thread. If that call were replaced with **`condition->broadcast(lock)`**, do you think that the procedures would still work correctly?

b. Describe what would happen when a broadcast took place. If you think it won't work correctly, describe an example where it fails. If you think it will work correctly, describe why it won't fail.

8. In the Synchronization Part 2 lecture, there is also an extensive example of a database reader/writer implementation.

a. Many readers can access the database at the same time. Describe a circumstance under which one or more readers can starve (ie, never get access to the database).

b. The procedure DoneWrite makes sure that there are no waiting writers before it wakes up the waiting readers with the call **`okToRead->Broadcast(lock)`**. This is the only place that readers are notified that it's okay to read.

Describe a circumstance in which the StartRead procedure, immediately after returning from **`okToRead->Wait(lock)`**, would find that there was a waiting writer even though DoneWrite had just checked this same condition before doing the broadcast.