CIS 415 * Spring 2010
Quiz 3: Synchronization Basics (25 pts)

Name: ________________________________ Score_________
(diagnostic only)

1. Reasons for synchronization (5 pts)
Check those that are feasible reasons for the OS to provide synchronization kernel calls.

_____ coordinate use of I/O buffers
_____ coordinate a parent process and a child process using a shared data structure
_____ coordinate a process printing to two different disks
_____ prevent data from getting overwritten before it has been read
_____ cause one process to begin execution after it has been signaled by another

2. Synchronization Solutions (6 pts)

a. List one disadvantage of using software solutions for synchronization of application programs.

b. Why must the code for the producer/consumer problem be synchronized?

c. What is the atomic test-and-set(target) hardware instruction used for?

3. Semaphore Facts (6 pts)

a. Which of the following is true of the semaphore operations wait and signal?

_____ are also specified by B() and W() -- blahooeyglugen (means “wait” in Dutch) and whohagglegligen (means “signal” in Dutch)
_____ must be executed atomically
_____ operate on binary semaphores or counting semaphores
_____ implemented with busywaiting version or blocking version.
_____ more error prone for use in mutual exclusion than software solutions
_____ are used for both application level and kernel level synchronization problems
4. Semaphore Usage (8 pts)

Consider 20 processes, P1, P2, ..., P20. Suppose the first ten processes are executing

P(mutex); critical section; V(mutex); /* semaphore mutex was initialized to 4 */

The second group of ten processes are executing;

P(cutex); critical section; V(cutex); /* semaphore cutex was initialized to 3 */

Assume the underlying OS kernel supports blocking implementation of P() and V(). Assume only these 20 processes are active in the system.

<table>
<thead>
<tr>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many total processes could be executing in their critical sections concurrently?</td>
<td></td>
</tr>
<tr>
<td>How many processes could be blocked on semaphore mutex?</td>
<td></td>
</tr>
<tr>
<td>If P11 and P12 only are in critical section, how many processes could be blocked on cutex?</td>
<td></td>
</tr>
<tr>
<td>How many processes could be executing V(mutex) code concurrently?</td>
<td></td>
</tr>
<tr>
<td>How many processes could be executing either P(mutex) or P(cutex) concurrently?</td>
<td></td>
</tr>
<tr>
<td>How many processes could be in the ready queue if one process is blocked on mutex and one process is blocked on cutex?</td>
<td></td>
</tr>
</tbody>
</table>

Assume the OS implemented semaphores with a busywaiting implementation (instead of blocking implementation). Would any of your answers above be different? _____ YES _____ No

Explain!