Name:

Operating Systems
Fall 2013
Test 2
October 28, 2013

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. No calculator needed.

You have 75 minutes to solve 8 problems. You get 10 points for writing your name on the top of this page. As with any exam, you should read through the questions first and start with those that you are most comfortable with. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

Partial credit will be offered only for meaningful progress towards solving the problems.

Please read and sign below if you agree with the following statement:

In recognition of and in the spirit of the academic code of honor, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:_____

0	/10
1	/10
2	/15
3	/15
4	/10
5	/10
6	/10
7	/10
8	/10
Total	**/100**

1. (10 points) **True or False?**
    a. It is possible in a single-processor system to not only interleave the execution of multiple processes but also to overlap their execution on the CPU.

    b. Race condition is a situation in which two or more processes continuously change their states in response to changes in the other process(es) without doing any useful work.

    c. Atomicity guarantees isolation from concurrent processes.

    d. A process that is waiting for access to a critical section protected by semaphores does not consume processor time.

    e. Deadlock avoidance requires knowledge of future process resource requests.

    f. An unsafe state is one in which there is at least one sequence of resource allocations to processes that does not result in a deadlock.

    g. Deadlock avoidance is more restrictive than deadlock prevention.

    h. The medium-term scheduler is invoked whenever an event occurs that may lead to the blocking of the current process or that may provide an opportunity to preempt a currently running process in favor of another.

    i. In a multiprogramming system multiple processes exist concurrently in main memory.

    j. In general we can claim that a resource allocation graph with no cycles shows that deadlock is impossible.

2. (15 points) **Short answers**:

a. In the case of competing processes three control problems must be faced: mutual exclusion, deadlock, and _____ .

b. A _____ is a semaphore that takes on only the values of 0 and 1.

c. A monitor supports synchronization by the use of _____ that are contained within the monitor and accessible only within the monitor.

d. Three general approaches exist for dealing with deadlock: prevent, avoid, and _____ .

e. The _____ condition for deadlock can be prevented by defining a linear ordering of resource types.

f. A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain is the condition of _____ .

g. Which of the following scheduling policies allow the O/S to interrupt the currently running process and move it to the Ready state? (choose one)

A. FIFO B. FCFS C. non-preemptive D. preemptive

h. A main characteristic of monitors is (choose one):

A. A maximum of two processes may be executing in a monitor at a time
B. Local data variables of the monitor are accessible by any procedure requesting use of the monitor
C. A process enters the monitor by invoking one of its procedures
D. All of the above

i. A risk with _____ is the possibility of starvation for longer processes, as long as there is a steady supply of shorter processes.

j. The _____ approach means that the operating system allocates the processor to a process and when the process blocks or is preempted, feeds it back into one of several priority queues.

3. (15 points) **Short attention span:** Answer the following questions (5 points each)
    a. What is the difference between deadlock and starvation?

    b. Suppose the following two processes, foo and bar are executed concurrently and share the semaphore variables S and R (each initialized to 1) and the integer variable x (initialized to 0).

```
void foo( ) {
    while (1) {
       wait(S);
       wait(R);
       x++;
       signal(S);
       signal(R);
   }
}
```

```
void bar( ) {
     while (1){
        wait(R);
        wait(S);
        x--;
        signal(S);
        signal(R);
     }
}
```

    Can the concurrent execution of these two processes result in one or both being blocked forever? If yes, give an execution sequence in which one or both are blocked forever.

    c. In the problem above, can the concurrent execution of the two processes result in the indefinite postponement of one of them? If yes, give an execution sequence in which one is indefinitely postponed.

4. (10 points) **Scheduling**: Is it possible for a CPU scheduler with a 100-millisecond time slice (quantum) to spend over half its time in the OS context switch code? Assume that it takes the OS 1 millisecond to context switch the CPU. Justify your answer.

5. (10 points) **Scheduling**: Suppose a scheduling algorithm favors those processes that have used little processor time in the recent past.

    a. Explain why this algorithm favors I/O-bound processes.
    b. Explain why this algorithm does not permanently deny processor time to CPU-bound processes.

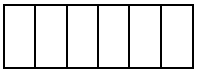
6. (10 points) **Synchronization**: Two threads are sharing a buffer as shown below. The buffer will be accessed by the threads from left to right in a **circular** fashion. Thread 2 reads characters from a file and places them in the buffer. Thread 1 reads characters from the buffer and prints them to the screen. The buffer is initialized with "blank" characters in each position. To protect the buffer from overwriting by thread 2 or overreading by thread 1, a confused programmer creates two semaphores, A and B and initializes A to 2 and B to 10 and places them in the threads as shown below. The content of the file is as follows:

abcdefghijklmnop

Thread 2 is assigned the CPU first and runs until it is blocked by the semaphore. Then Thread 1 runs until it is blocked by the semaphore.

Thread 1	**Thread 2**
Repeat wait(A) read item from buffer print item to screen signal(B) until false	Repeat wait(B) read character from file place item in buffer signal(A) until false

Describe the content of the buffer when Thread 2 was blocked and explain why the buffer contains that.



Describe what Thread 1 printed to the screen and explain why it printed that.

7. (10 points) **Deadlock**: Consider a system consisting of four processes and a single resource. The current state of the claim and allocation matrices are:

   C = (3 2 9 7)

   A = (1 1 3 2)

   What is the minimum number of units of the resource needed to be available for this state to be safe?

8. (10 points) **Producer-Consumer:** Below is the correct solution to the producer/consumer problem using semaphores. Consider the following modification: the semWait() operations in the producer (shown by the arrow) are inversed. That is, consider that semWait(s) is now before semWait(e). What will happen in this case?

```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore s = 1, n= 0, e= sizeofbuffer;
void producer()
{
     while (true) {
          produce();
          semWait(e);
          semWait(s);
          append();
          semSignal(s);
          semSignal(n);
     }
}
void consumer()
{
     while (true) {
          semWait(n);
          semWait(s);
          take();
          semSignal(s);
          semSignal(e);
          consume();
     }
}
void main()
{
     parbegin (producer, consumer);
}
```