MCS-378 Intraterm Exam 1

Serial #:

This exam is closed-book and mostly closed-notes. You may, however, use a single 8 1/2 by 11 sheet of paper with *hand-written* notes for reference. (Both sides of the sheet are OK.)

Please write your name only on this page. Do not turn the page until instructed, in order that everyone may have the same time. Then, be sure to look at all problems before deciding which one to do first. Some problems are easier than others, so plan your time accordingly. You have 50 minutes to work.

Write the answer to each problem on the page on which that problem appears. You may also request additional paper, which should be labeled with your test number and the problem number.

Printed name: ______________________________

On my honor, I pledge that I have not given, received, nor tolerated others' use of unauthorized aid in completing any work for this course.

Signature for above honor pledge: ______________________________

Problem	Page	Possible	Score
1	2	20	
2	3	20	
3	4	20	
4	5	20	
5	6	20	
Total		100	

1. [**20 Points**] Suppose thread T1 is triggered by an event at time 0, and needs to run for 10 seconds before it can respond. Suppose thread T2 is triggered by an event occuring 3 seconds later than T1's trigger, and that T2 needs to run 5 seconds before it can respond. Draw a Gantt chart for each of the following three cases, and for each indicate the response time of T1, the response time of T2, and the average response time:

   (a) T1 is allowed to run to completion before T2 is run.

   (b) T1 is preempted when T2 is triggered; only after T2 has completed does T1 resume.

   (c) T1 is preempted when T2 is triggered; the two threads are then executed in a round-robbin fashion (starting with T2), until one of them completes. The time slice (or quantum) is 1 second.

2. [**20 Points**] Consider a uniprocessor machine running a Unix-family system such as Mac OS X or Linux. (Recall that OS X uses a decay usage scheduler.) Two single-threaded processes are running with different niceness levels.

   (a) Suppose both processes are carrying out long running, purely CPU-bound computations. Which receives a greater proportion of the CPU time: the one with a numerically greater niceness or the one with the lesser niceness?

   (b) On the OS X system, each process has a priority as well as a niceness. Briefly explain why the priority is not the same as the niceness.

   (c) Continuing with the assumption of long running, purely CPU-bound computations, the relative proportion of CPU time each receives can adjusted by adjusting the difference in niceness levels. Suppose someone has a target ratio of CPU shares they want to achieve. Explain why the corresponding difference in niceness levels would be simpler to calculate for Linux than for Mac OS X.

   (d) Suppose that one of the processes is largely I/O bound, doing little computation, while the other remains CPU-bound. Explain why with the standard Linux scheduling policy, the I/O bound process's performance will not be significantly slowed by sharing the CPU with the CPU-bound process, even if the CPU-bound process has a niceness advantage.

   (e) Continuing with the assumption that one of the processes is largely I/O bound, doing little computation, while the other is CPU-bound, turn your attention now to OS X. Explain why the niceness difference between the processes determines whether the I/O bound process's performance will be significantly slowed by sharing the CPU with the CPU-bound process.

3. [**20 Points**] The following Semaphore class is missing some code at the beginning of the down method:

```
public class Semaphore {
    private int value;

    public Semaphore(int value) {
        this.value = value;
    }

    public synchronized void up() {
        value++;
        notifyAll();
    }

    public synchronized void down()
        throws InterruptedException
    {






        value--;
    }
}
```

   (a) Insert the missing code.

   (b) Could you change notifyAll to notify in the up method without sacrificing correctness? Explain why or why not. (Keep in mind that in general, there are two different reasons why a replacement of notifyAll by notify might be incorrect.)

4. [**20 Points**]

   (a) A semaphore can be used as a mutex. Which of the two semaphore operations, `down` and `up`, would be used to lock the mutex, and which would be used to unlock it?

   (b) If each account at a bank is protected by a mutex, the code to transfer money from one account to the other might look something like this:

   - lock the source account's mutex
   - lock the destination account's mutex
   - update the source account
   - update the destination account
   - unlock the source account's mutex
   - unlock the destination account's mutex

   However, with this approach, concurrent transfers could deadlock. How should the code be changed to prevent deadlock?

   (c) Instead of preventing deadlock, it is possible to detect it once it occurs, abort one of the transfers, and later retry it. This approach might be somewhat easier with the above code than it would be with the following variant. Explain why.

   - lock the source account's mutex
   - update the source account
   - lock the destination account's mutex
   - update the destination account
   - unlock the source account's mutex
   - unlock the destination account's mutex

   (d) Does the original version of the code obey two-phase locking? How about the variant given in the previous subproblem; does it obey two-phase locking? Explain your answers.

5. [**20 Points**] For each of the following histories, if the history is serializable, give an equivalent serial history. Rather than listing all the steps in the serial history, you can just list the transaction numbers (1 and 2; or 1, 2, and 3) in the appropriate order. If the history is not serializable, say so. If the history is serializable in more than one order, list all legal orders.

   (a) $s_1(x), r_1(x), \overline{s}_1(x), s_2(y), r_2(y), \overline{s}_2(y), s_1(y), r_1(y), \overline{s}_1(y), e_2(x), w_2(x), \overline{e}_2(x)$

   (b) $s_1(x), r_1(x), \overline{s}_1(x), e_2(y), w_2(y), \overline{e}_2(y), s_1(y), r_1(y), \overline{s}_1(y), e_2(x), w_2(x), \overline{e}_2(x)$

   (c) $s_1(x), r_1(x), \overline{s}_1(x), s_2(y), r_2(y), \overline{s}_2(y), s_1(y), r_1(y), \overline{s}_1(y), s_2(x), r_2(x), \overline{s}_2(x)$

   (d) $s_1(x), r_1(x), \overline{s}_1(x), e_2(y), w_2(y), \overline{e}_2(y), s_1(y), r_1(y), \overline{s}_1(y), s_2(x), r_2(x), \overline{s}_2(x)$

   (e) $e_1(x), w_1(x), \overline{e}_1(x), e_2(y), w_2(y), \overline{e}_2(y), e_3(x), w_3(x), \overline{e}_3(x), e_3(y), w_3(y), \overline{e}_3(y)$