# 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 A goes through a loop 100 times, each time performing one disk I/O operation, taking 9 ms, and then some computation, taking 3 ms. While each 9 ms disk operation is in progress, thread A cannot make any use of the processor. Thread B runs for 1 second, purely in the processor, with no I/O. One millisecond of processor time is spent each time the processor switches threads; other than this switching cost, there is no problem with the processor working on thread B during one of thread A's I/O operations. (The processor and disk drive do not contend for memory access bandwidth, for example.) Give your answers to the following questions in milliseconds.

   (a) Suppose the processor and disk work purely on thread A until its completion, then the processor switches to thread B and runs all of that thread. What will the total elapsed time be?

   (b) Suppose the processor starts out working on thread A, but every time thread A performs a disk operation, the processor switches to B during the operation, then back to A upon the disk operation's completion. What will the total elapsed time be?

2. [ **20 Points** ] Suppose thread T1 is triggered by an event at time 0 and needs to run for 4 seconds before it can respond. Suppose thread T2 is triggered by an event occuring later than T1's trigger, and that T2 needs to run 2 seconds before it can respond. Draw a Gantt chart for each of the following four cases, and for each indicate the response time of T1, the response time of T2, and the average response time:

   (a) T2 is triggered by an event that is 1 second later than T1's trigger; T1 is allowed to run to completion before T2 is run.

   (b) T2 is triggered by an event that is 1 second later than T1's trigger; T1 is preempted when T2 is triggered; only after T2 has completed does T1 resume.

   (c) T2 is triggered by an event that is 3 seconds later than T1's trigger; T1 is allowed to run to completion before T2 is run.

   (d) T2 is triggered by an event that is 3 seconds later than T1's trigger; T1 is preempted when T2 is triggered; only after T2 has completed does T1 resume.

   Generalizing from cases (a)-(d), we can imagine situations where T2 would be triggered at some other time during T1's execution (not just at 1 second or 3 seconds). We could even further generalize and consider execution times for T1 and T2 other than 4 and 2 seconds. State a general rule for when the average response time is minimized by allowing T1 to continue running and when it is minimized by preempting T1 and running T2.

3. [ **20 Points** ]

(a) Complete the following Java class, `Resource`, which is to be used in a multi-threaded program to represent the available quantity of some resource. Each newly created resource has a quantity of 0. The `change` operation can be used to change the quantity; if the argument (`delta`) is positive, the quantity goes up by the specified amount. If the argument is negative, it still specifies how much the quantity should change by, but this now naturally means that the quantity goes down. You are to complete the code in such a way that any thread performing a `change` operation is temporarily suspended if necessary to ensure that the quantity never goes negative.

```
public class Resource {
    private int quantity = 0;

    public synchronized void change(int delta)
                                throws InterruptedException
    {




        quantity = quantity + delta;
        notifyAll();
    }
}
```

(b) What modification could be made in the code I provided that might improve its efficiency without hurting its correctness?

4. [ **20 Points** ] Write down five different two-phase histories that would be possible for a transaction, $T_1$, that first reads $x$, then writes $y$, then writes $z$. In at least two of your five histories, avoid grouping all the unlock operations together at the end of the history.

5. [ **20 Points** ] Give an example of a transaction where it matters that undo log entries are processed in reverse chronological order and explain why it matters.