CS3331 Concurrent Computing Exam 1
Spring 2014

100 points – 6 pages

Name: _____

- Most of the following questions only require short answers. Usually a few sentences would be sufficient. Please write to the point. If I don't understand what you are saying, I believe, in most cases, you don't understand the subject.
- To minimize confusion in grading, please write **readable** answers. Otherwise, it is very likely I will interpret your unreadable handwriting in my way.
- *Justify your answer with a convincing argument*. An answer **must** include a convincing justification. You will receive no point for that question even though you have provided a correct answer. *I consider a good and correct justification more important than just providing a right answer. Thus, if you provide a very vague answer without a convincing argument to show your answer being correct, you will likely receive a low to very low grade.*
- You **MUST** use execution sequences to answer synchronization related problem. Without doing so, you risk a lower grade.
- Note that prove by an example (*i.e.*, using an example as a proof) is NOT a proof!
- Do those problems you know how to do first. Otherwise, you may not be able to complete this exam on time.

1. **Basic Concepts**

   (a) **[10 points]** Explain *interrupts* and *traps*, and provide a detailed account of the procedure that an operating system handles an interrupt.

   (b) **[10 points]** What is an atomic instruction? What would happen if multiple CPUs/cores execute their atomic instructions?

2. **Processes**

   (a) **[10 points]** What is a *context*? Provide a detail description of *all* activities of a *context switch*.

(b) **[10 points]** Draw the state diagram of a process from its creation to termination, including all transitions. Make sure you will elaborate **every state** and **every transition** in the diagram.

3. **Threads**

(a) **[10 points]** Why is handling threads cheaper than handling processes? **Provide an accurate account of your findings. Otherwise, you risk a lower grade.**

4. **Synchronization**

(a) **[10 points]** Define the meaning of a *race condition*? Answer the question first and use an execution sequence with a clear and convincing argument to illustrate your answer. **You must explain step-by-step why your example causes a race condition.**

(b) **[10 points]** A computer system has two CPUs that share the same memory. All processes are stored in the shared memory but can be run on either CPU. To gain efficiency, the designers chose the following CPU scheduling policy:

- There is only one ready queue and is stored in the shared memory.
- Each CPU has its own CPU scheduler.
- When a CPU is free, the scheduler of that CPU picks up the first process in the ready queue to run on the same CPU.

Do you think this policy works well? State your claim first and justify your claim step-by-step with an execution sequence. *You will receive* **no** *credit if you only provide an answer* **without** *a convincing argument or if your answer is* **vague**.

5. **Problem Solving:**

(a) **[15 points]** Consider the following two processes, A and B, to be run concurrently using a shared memory for variable `x`.

```
Process A                      Process B
---------                      ---------
for (i = 1; i <= 2; i++)       x = 2*x;
   x++;
```

Assume that load and store of `x` is atomic, `x` is initialized to 0, and `x` must be loaded into a register before further computations can take place. What are all possible values of `x` after both processes have terminated. Use a step-by-step execution sequence of the above processes to show all possible results. **You must provide a clear step-by-step execution of the above algorithm with a convincing argument. Any vague and unconvincing argument receives no points.**

Use this and next page for your answer.

(b) **[15 points]** Consider the following solution to the mutual exclusion problem for two processes P_1 and P_2. This solution uses two global `int` variables, `x` and `y`. Both `x` and `y` are initialized to `0`.

```
int x = 0, y = 0;
```

Process 1

```
START:
     x = 1;
     if (y != 0) {
          repeat until (y == 0);
          goto START;
     }
     y = 1;
     if (x != 1) {
          y = 0;
          repeat until (x == 0);
          goto START;
     }
// critical section
     x = y = 0;
```

Process 2

```
START:                              // All start from here
     x = 2;                         // set my ID to x
     if (y != 0) {                  // if y is non-zero
          repeat until (y == 0);    //    wait until y = 0
          goto START;               //    then try again
     }                              // second section
     y = 1;                         // set y to 1
     if (x != 2) {                  // if x is not my ID
          y = 0;                    //    set y to 0
          repeat until (x == 0);    //    wait until x = 0
          goto START;               //    then try again
     }
// critical section
     x = y = 0;                     // set x and y to 0
```

Prove rigorously that this solution satisfies the mutual exclusion condition. *You will receive* **zero** *point if* **(1)** *you prove by example, or* **(2)** *your proof is vague and/or unconvincing.*

Grade Report

Problem		*Possible*	*You Received*
1	a	10	
	b	10	
2	a	10	
	b	10	
3	a	10	
4	a	10	
	b	10	
5	a	15	
	b	15	
Total		100	