CS3331 Concurrent Computing Solution 1 – Spring 2015

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] Explain the one-to-one, many-to-one, and many-to-many thread models. Make sure you explain each model clearly.

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.

5. Problem Solving:

(a) [10 points] Consider the following program segment. Suppose all fork() calls are successful.

Answer the following questions: (1) Draw a diagram showing the parent-child relationship of all

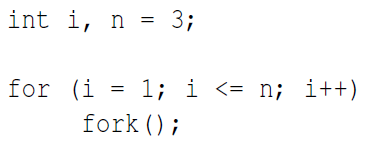
involved processes, the main program included, and provide an explanation how this relationship

is obtained. Vague and not convincing arguments receive zero point. (2) This program

segment uses n = 3. How many processes will be created if n is set to a positive integer k?

You don’t need to draw a diagram; but you still have to provide a justification for your answer.

Vague and not convincing argument receive zero point.



(b) [15 points] Consider the following two processes, A and B, to be run concurrently using a shared

memory for variable x.



Assume that 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.

(c) [15 points] Consider the following solution to the mutual exclusion problem for two processes P0 and P1. A process can be making a request REQUESTING, executing in the critical section IN CS, or having nothing to do with the critical section OUT CS. This status information, which is represented by an int, is saved in flag[i] of process Pi. Moreover, variable turn is initialized elsewhere to be 0 or 1. Note that flag[] and turn are global variables shared by both P0 and P1.

