CSCI 40300/ECE 40800
Operating Systems– Fall 2016
Quiz 4
Solutions

Name: _____

Page:	1	2	Total
Points:	10	10	20
Score:			

Normalized Total to $100 = 100 \times \text{Total}/20 =$ _____ (what will appear in Canvas gradebook).

1. (10 points) 5 identical jobs are run once on a non-preemptive scheduler, and then again on a preemptive scheduler using a round robin scheduling discipline. The jobs are purely computational — they do almost no I/O. On the non-preemptive scheduler it takes 24 hours for all 5 of them to complete; on the preemptive one it takes 24 hours and 2 minutes for all 5 of them. If the time quantum of the preemptive scheduler is 0.05 sec (i.e., 50 milliseconds), how long does a context switch take? (Show all your work; it's OK to leave the answer in symbolic / long form).

   Answer: We are told that all the processes are identical and CPU-bound. Therefore, with the non-preemptive scheduler, the context switches occur when the processes CPU bursts finish. With time quantum of 50 milliseconds, there are 20 context switches per second for the RR scheduler. We are told in the problem that the total time for this overhead over a 24 hour period is 2 minutes. Over the 24 hour period there are 24 hours $\times$ 60 min/hr $\times$ 60 sec/min $\times$ 20 switches/sec = 1728000 context switches. These take a total of 120 seconds. Each context switch then takes 120 / 1728000 sec = 1/14400 sec = 69.4 microseconds.

Name: _____

2. Consider the multi-level feedback queue based scheduling of processes with demotions and promotions up and down the different levels of queues with different time quanta.

   (a) (5 points) It has been claimed by some that a multi-level feedback queue scheduler approximates Shortest Remaining Time First (SRTF). Give an explanation why this could be true.

   Answer: If the time quantum expires, drop the thread one queue level, else push the thread up one queue level. Using this algorithm, short-running CPU jobs stay near the top, and CPU-bound jobs drop toward the bottom. Assuming that the high queue levels are given more priority than the lower ones, this tends to approximate SRTF because it gives more CPU cycles to jobs with short bursts.

   (b) (5 points) Suggest a way *to fool* the multi-level feedback queue scheduler's heuristics described in part (a) into giving a long-running task higher priority to run on the CPU.

   Answer: The key word here is "to fool." The trick is to try to shorten the CPU bursts of the long-running task. We can do this by periodically doing I/O (such as a bunch of print statements) that doesn't change the computation itself. This way a given long-running process can keep its burst-length short and hence its priority high in the multi-level feedback scheduler.