# Homework 2

**Due Date:** October 30, 2008 **Points:** 100

## Questions

1. (*10 points*) Consider the disk head scheduling algorithms for a personal computer.
   (a) When disk I/O requests are arriving infrequently enough so that each request can be serviced before the next request arrives, to which disk head scheduling algorithm to *all* the algorithms degenerate?
   (b) In light of your answer, are disk scheduling policies other than the one named in the answer to 1a useful when at most one user at a time will be using the computer?

   Remember to justify your answer.

2. (*15 points*) Why is rotational optimization not usually employed in disk scheduling? How would the SSTF and SCAN (or LOOK) algorithms be modified to include rotational optimization?

3. (*20 points*) The clock interrupt handler on a certain computer requires 1 millisecond (including process switching overhead) per clock tick. The clock runs at 60 Hz (that is, it ticks 60 times per second). What fraction of the processor is devoted to the clock? (*text*, §3, problem 31, modified)

4. (*30 points*) Measurements of a certain system have shown that the average process runs for a time $t$ before blocking on I/O. A process switch requires a time $s$, which is effectively wasted (i.e., it is overhead time). For round robin scheduling with a quantum $q$, give a formula for the processor utilization for each of the following:
   (a) $q = \infty$
   (b) $q >= t$
   (c) $s < q < t$
   (d) $q = s$
   (e) $q \approx 0$

   (*text*, §3, problem 26)

5. (*25 points*) Suppose a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes which have used little processor time in the recent past. Why will this algorithm favor I/O bound processes and yet not permanently starve CPU-bound processes?

## Extra Credit

1. (*20 points*) The CDC 6600 computers could handle up to 10 I/O processes simultaneously using an interesting form of round robin scheduling called *processor sharing*. A process switch occurred after each instruction, so instruction 1 came from process 1, instruction 2 came from process 2, and so on. The process switching was done by special hardware, and the overhead was zero.
   (a) If a process needed $t$ seconds to complete in the absence of competition, how much time would it need if processor sharing was used with $n$ processes? What would the response ratio be in this case? (*text*, §2, ex. 24, modified)
   (b) Please reconcile your answer with problem 4e. Why are the two not contradictory?