CS 162 – Section 4

1) Scheduling techniques

Assumptions: All timeslice-based algorithms have a timeslice of one unit; The currently running thread is not in the ready queue while it is running; An arriving thread is run at the beginning of its arrival time, if the scheduling policy allows it. Turnaround time is defined as the time a process takes to complete after it arrives.

Fill in ALL blanks in EACH table – each blank has an unambiguous answer.
For the missing schedulers, the possibilities are **SRTF, RR, and Priority.**
Priority is a preemptive scheduler.
Hint: Fill in the entry time (below) for Thread C first!

Entry Times	
A	1
B	2
C	5
D	8

Priorities	
A	3
B	4
C	5
D	6

↓Current Time	Currently Scheduled Process		
Scheduler →	FIFO	SRTF	Priority
1	A	A	A
2	A	A	B
3	A	A	B
4	B	B	B
5	B	C	C
6	B	B	A
7	C	B	A
8	D	D	D
9	D	D	D
10	D	D	D
Avg Turnaround Time	3.5	3.25	3.5

2) Short-answer scheduling potpourri

a. Give two ways in which to predict runtime in order to approximate SRTF:
   Two options that we gave in class were (1) use a predictive algorithm that uses past behavior to predict the future (such as a weighted average of the previous prediction and the previous value), (2) use a scheduling algorithm(such as a multi-level scheduler) that tries to separate short burst applications from long burst applications and schedules the short-burst applications with higher priority.

b. What scheduling problem did the original Mars rover experience? What were the consequences of this problem?
   The original Mars rover exhibited a priority inversion problem: A low-priority task grabbed a lock on the bus; a high-priority task was forced to wait for the bus; further, a medium priority task prevented the low-priority task from completing. The result was that the high-priority task made no forward progress; eventually, a timer went off, decided something was wrong, and rebooted the system. Consequently, the system kept rebooting.

c. Five jobs are waiting to be run. Their expected running times are 10, 8, 3, 1, and X. In what order should they be run to minimize average completion time? State the scheduling algorithm that should be used AND the order in which the jobs should be run. HINT: Your answer will explicitly depend on X.
   To minimize average completion time, we need to use SRTF. The answer depends on X in the following way:
   X, 1, 3, 8, 10 if $X < 1$
   1, X, 3, 8, 10 if $1 \leq X < 3$
   1, 3, X, 8, 10 if $3 \leq X < 8$
   1, 3, 8, X, 10 if $8 \leq X < 10$
   1, 3, 8, 10, X if $10 \leq X$

d. Can any of the three scheduling schemes (FCFS, SRTF, or RR) result in starvation? If so, how might you fix this?
   Yes. SRTF continuously places short jobs in front of long jobs. Consequently, it is possible for long jobs to never get a chance to run.

   You can fix this by utilizing an algorithm that guarantees that no thread is prevented from making forward progress. Example: use a lottery scheduler or a multi-level scheduler that gives a guaranteed minimum amount of compute time to the lowest-priority (background) tasks.

   Note that FCFS and RR don't really experience starvation (although you might be able to make an argument that RR has a problem if you always put new jobs at the front of the queue and short jobs arrive faster than the quantum).

e. Explain why a chess program running against another program on the same machine might want to perform a lot of superfluous I/O operations.

By executing a lot of superfluous I/O, the chess program might be able to fool a scheduler into thinking that the chess program is an interactive task that should get more compute cycles. The net effect would be that the program performing the superfluous I/O might be able to get more “thinking” done per unit time that the one that didn’t perform extra I/O.