Process Scheduling Exercise

Below are 10 questions pertaining to Problem 9.1, page 428, Stallings textbook 8th edition (Problem 9.2, page 427, of 7th edition; earlier editions it was Problem 9.1 p426). Please submit answers to them onto the remote Unix machine using these preparation and submittal instructions. Please name your file "scheduling".

Problem 9.2 asks you to duplicate the analysis found in the book's Table 9.5 and Figure 9.5, but using different input values (the problem's instead of Table 9.4's). For purposes of this assignment, please do the problem for these four scheduling policies only:

- first-come first-served (FCFS)
- round robin, quantum=1 (RR1)
- shortest process next (SPN)
- shortest remaining time (SRT)

Prepare the equivalent of Figure 9.5 and Table 9.5 for those four policies, using Problem 9.2's input values. Then answer this assignment's ten questions, given below.

I performed the exercise of doing this for Table 9.4's input values (those used in the book discussion). I used an Excel spreadsheet as a tool, as opposed to doing it on paper. The spreadsheet is available, if you would like to use it. You can get it from the remote Unix machine. The name of the spreadsheet file is process-scheduling.xls. Alternatively, get it right here, in the form of a zip file (unzip to obtain the xls spreadsheet file).

Remember that FCFS and SPN are both "non-preemptive" policies. That means that once any process starts, you let it run to conclusion. Choosing which process will run next is not performed every time slice. Only on those slices right after the running process has finished. With RR1 and SRT, on the other hand, every time slice is occasion for a new decision.

As you progressively build the queue, whenever the running process is "bumped" but hasn't finished, put it at the very end of the queue behind any processes that were already in the queue. (While, for those processes, preserving their order.) If a new process enters, the same applies, it goes behind existing queued processes. On any time slice that has *both* a bumped-but-unfinished *and* a brand new process, the latter goes ahead of the former. So on such occasions the new queue will consist of 1) whatever processes were in the queue already, followed by 2) the brand new process, followed by 3) the one that just got bumped. When a process finishes (i.e., after it has been given its "service time" number of slices) remember not to inject it back into the queue; just record its termination (which is an important input to the table you will construct). The queue is only used on the occasions when you are going to pick a new process to run; for the non-preemptive cases don't try to apply the queue every slice, but keep maintaining it so you will have it next time you do need to apply it.

Assignment:

Refer to the table you produced based on Problem 9.2 as described above.

1. The scheduling policy for which process D finishes the latest is
a. FCFS
b. RR1
c. SPN
d. SRT

2. The scheduling policy for which process B finishes the earliest is
a. FCFS
b. RR1
c. SPN
d. SRT

3. The process that finishes at the same time regardless of scheduling policy is
a. A
b. B
c. C
d. D
e. E

4. The longest residency (or "turnaround") time T_r for any process under any scheduling policy is
a. 8
b. 10
c. 15
d. 18
e. 20

5. The poorest (greatest) normalized residency T_r/T_s for any process under any scheduling policy is
a. 1.98
b. 10.00
c. 20.00
d. 3.50
e. 6.20

6. The scheduling policy that exhibits the poorest average normalized residency time (averaged over all the processes) is
a. FCFS
b. RR1
c. SPN
d. SRT

7. The process that needs the smallest amount of execution time in order to run from start to finish (independent of scheduling algorithm) is
a. A
b. B

c. C
d. D
e. E

8. The scheduling policy for which the termination of the first process to complete occurs latest is
a. FCFS
b. RR1
c. SPN
d. SRT

9. If the normalized residency T_r/T_s of a process is 1 it means
a. The amount of time the process resided in the system was the same as that of all other processes
b. The process entered the system at the same time as its predecessor left the system
c. The amount of time the process resided in the system was the same as that of its predecessor
d. The amount of time the process resided in the system was the same as that of its successor
e. The amount of time the process resided in the system was kept to a minimum

10. The scheduling policy for which the execution of processes is most highly interspersed (intermixed instead of one-at-a-time) is:
a. FCFS
b. RR1
c. SPN
d. SRT