Answer #1

a. The four Gantt charts are

FCFS

1	2	3	4	5

RR

1	2	3	4	5	1	3	5	1	5	1	5	1	5	1

SJF

2	4	3	5	10

Priority

2	5	10	3	4

b. Turnaround time

	FCFS	RR	SJF	Priority
P1	10	19	19	16
P2	11	2	1	1
P3	13	7	4	18
P4	14	4	2	19
P5	19	14	9	6

c. Waiting time (turnaround time – burst time)

	FCFS	RR	SJF	Priority
P1	0	9	9	6
P2	10	1	0	0
P3	11	5	2	16
P4	13	3	1	18
P5	14	9	4	1

d. Shortest Job First

Answer #2

This algorithm is called Dekker's solution.

1. **Mutual exclusion:**
   There are two cases to consider:
   a) **A process is inside the C.S.:** Without loss of generality, assume process j is inside the C.S. Before entering the C.S. the process sets its own flag to 1. If process i tries to enter the C.S. it will see that flag[j] is up and gets caught up in the while loop. It will continue in the while loop until the other process sets its own flag to 0, which happens only at the end of the C.S.
   b) **Two processes are trying to enter simultaneously:** In this situation, if both processes reach their respective while loop at the top, then the variable turn will ensure that only one of them passes through. The variable turn is alternating between the allowing either process, and is only modified at the exit of a C.S.
2. **Progress:**
   There are two cases to consider:
   a) **One process is trying to enter with no competition**: In such a case, the flag of the other process is down, and the process goes past the while look into the critical section directly.
   b) **Two processes are trying to enter simultaneously**. In this case if the first process is trapped into the while loop, then the variable turn will make one of the two variables lower its flag and goes into a loop waiting for the variable turn to change (the inner while loop). The other process whose turn is set by the variable turn will be able to get through.
3. **Bounded Waiting:**
   Assume there is a process blocked inside the inner while loop, while another process is in C.S. In such a case, if the process inside the critical section tries to re-enter, it will be blocked because on exit of the C.S. it has already set the variable turn to point to the other process. Therefore, the process that just got out of the C.S. will be forced to wait for its own turn. So, bounded waiting is taken care of.