<u>**Solutions to Week 1 Exercises**</u>

1. n! (n factorial)

2. Pre-emptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive Scheduling ensures that a process relinquishes control of the CPU only when it finishes its current CPU burst.

3. Solutions are:
   (a) 10.53
   (b) 9.53
   (c) 6.86

4. Processes that need more frequent servicing, for instance, interactive processes such as editors, can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing, and thus making more efficient use of the computer.

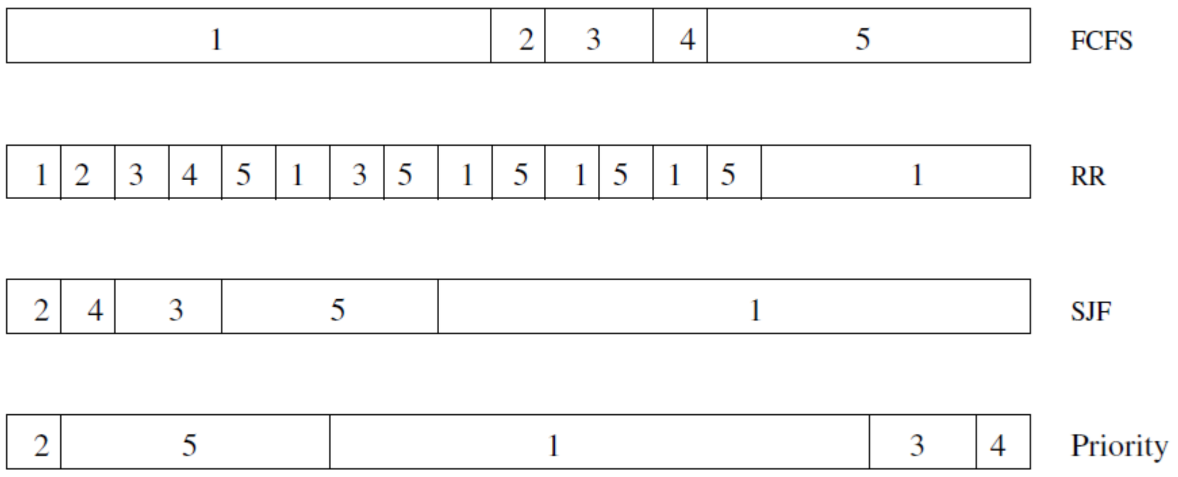
5. Solutions are:
   (a) The shortest job has the highest priority.
   (b) The lowest level of MLFQ is FCFS.
   (c) FCFS gives the highest priority to the job having been in existence the longest.
   (d) None.

6. It will favour the I/O -bound programs because of the relatively short CPU burst request by them; however, the CPU -bound programs will not starve because the I/O -bound programs will relinquish the CPU relatively often to do their I/O .

7. I/O-bound programs have the property of performing only a small amount of computation before performing I/O. Such programs typically do not use up their entire CPU quantum. CPU-bound programs, on the other hand, use their entire quantum without performing any blocking IO operations. Consequently, one could make better use of the computer's resources by giving higher priority to I/O-bound programs and allow them to execute ahead of the CPU-bound programs.

8.
   a. *CPU utilization and response time*: CPU utilization is increased if the overheads associated with context switching is minimized. The context switching overheads could be lowered by performing context switches infrequently. This could however result in increasing the response time for processes.
   b. *Average turnaround time and maximum waiting time*: Average turnaround time is minimized by executing the shortest tasks first. Such a scheduling policy could however starve long-running tasks and thereby increase their waiting time.
   c. I/O device utilization and CPU utilization: CPU utilization is maximized by running long-running CPU-bound tasks without performing context switches. I/O device utilization is maximized by scheduling I/O-bound jobs as soon as they become ready to run, thereby incurring the overheads of context switches.

9.
   (a) Gannt Charts:



(b) Turnaround Times:

	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 Times

	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.

10. Shortest Job First and Priority.

11.
(a) In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment.
(b) The advantage is that more important jobs could be given more time, in other words, higher priority in treatment. The consequence, of course, is that shorter jobs will suffer.
(c) Give a longer amount of time to processes deserving higher priority. In other words, have two or more quanta possible in the Round-Robin scheme.

12. Answer:
(a) The time quantum is 1 millisecond: Irrespective of which process is scheduled, the scheduler incurs a 0.1 millisecond context-switching cost for every context-switch. This

results in a CPU utilization of 1/1.1 * 100 = 91%.

(b) The time quantum is 10 milliseconds: The I/O-bound tasks incur a context switch after using up only 1 millisecond of the time quantum. The time required to cycle through all the processes is therefore 10*1.1 + 10.1 (as each I/O-bound task executes for 1 millisecond and then incur the context switch task, whereas the CPU-bound task executes for 10 milliseconds before incurring a context switch). The CPU utilization is therefore 20/21.1 * 100 = 94%.

13. The program could maximize the CPU time allocated to it by not fully utilizing its time quantums. It could use a large fraction of its assigned quantum, but relinquish the CPU before the end of the quantum, thereby increasing the priority associated with the process.

14.
    (a) FCFS
    (b) LIFO (Last in First out)

15. Answer:
    (a) FCFS —discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.
    (b) RR —treats all jobs equally (giving them equal bursts of CPU time) so short jobs will be able to leave the system faster since they will finish first.
    (c) Multilevel feedback queues—work similar to the RR algorithm—they discriminate favourably toward short jobs.

16. Answers:
    (a) 26
    (b) 8
    (c) 14