5.11) Of these two types of programs:

a. I/O-bound
b. CPU-bound

which is more likely to have voluntary context switches, and which is more likely to have nonvoluntary context switches?

Answer:

a. I/O-bound—Voluntary, as the program is more likely to voluntarily relinquish the CPU as it waits for I/O to become available.

b. CPU-bound—Nonvoluntary, as the program is likely to use the CPU for its entire time quantum

5.16) A variation of the round-robin scheduler is the regressive round-robin scheduler. This scheduler assigns each process a time quantum and a priority. The initial value of a time quantum is 50 milliseconds. However, every time a process has been allocated the CPU and uses its entire time quantum (does not block for I/O), 10 milliseconds is added to its time quantum, and its priority level is boosted. (The time quantum for a process can be increased to a maximum of 100 milliseconds.) When a process blocks before using its entire time quantum, its time quantum is reduced by 5 milliseconds, but its priority remains the same. What type of process (CPU-bound or I/O-bound) does the regressive round-robin scheduler favor? Explain

Answer: This scheduler would favor CPU-bound processes, as they are rewarded with a longer time quantum as well as a priority boost whenever they consume an entire time quantum. This scheduler does not penalize I/O-bound processes, as they are likely to block for I/O before consuming their entire time quantum, but their priority remains the same.

5.18) The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm: Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. The scheduler will execute the highest-priority process. For processes with the same priority, a round-robin scheduler will be used with a time quantum of 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a. Show the scheduling order of the processes using a Gantt chart.
b. What is the turnaround time for each process?
c. What is the waiting time for each process?

Process	Priority	Burst	Arrival
P1	8	15	0
P2	3	20	0
P3	4	20	20

P4	4	20	25
P5	5	5	45
P6	5	15	55

Answer: a.

time=0	P1
time=15	P2
time=20	P3
time=30	P4
time=40	P3
time=45	P5
time=50	P3
time=55	P6
time=70	P4
time=80	P2

b. P1 = 15, P2 = 95, P3 = 35, P4 = 55, P5 = 5, P6 = 15

c. P1 = 0, P2 = 75, P3 = 15, P4 = 35, P5 = 0, P6 = 0

5.21) Consider a variant of the RR scheduling algorithm in which the entries in the ready queue are pointers to the PCBs.

a. What would be the effect of putting two pointers to the same process in the ready queue?

b. What would be two major advantages and two disadvantages of this scheme?

c. How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

Answer:
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. The consequence, of course, is that shorter jobs will suffer.

c. Allocate a longer amount of time to processes deserving higher priority. In other words, have two or more quanta possible in the round-robin scheme.

5.23) Consider a system implementing multilevel queue scheduling. What strategy can a computer user employ to maximize the mount of CPU time allocated to the user's process?

Answer: The program could maximize the CPU time allocated to it by not fully utilizing its time quanta. 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.

5.25) Explain how the following scheduling algorithms discriminate either in favor of or against short processes:

a. FCFS
b. RR
c. Multilevel feedback queues

Answer:
a. FCFS discriminates against short jobs since any short job arriving after a long job 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 similarly to the RR algorithmically they discriminate favorably toward short jobs.

5.27) Consider a load-balancing algorithm that ensures that each queue has approximately the same number of threads, independent of priority. How effectively would a priority-based scheduling algorithm handle this situation if one run queue had all high-priority threads and a second queue had all low-priority threads?

Answer: In this situation, the low-priority threads would get approximately the same amount of CPU time as the high-priority threads. This would not accurately reflect a priority-based scheduling algorithm, which is supposed to favor high-priority threads.