Question 1:

We have been benchmarking the programs being run on a server and have learned that we can easily classify all of the programs into three categories. Furthermore, we will know which category each program belongs to when it begins to execute:

- Interactive (I/O-bound) jobs with a short average CPU burst time of SBurst.
- Interactive (I/O-bound) jobs with a long average CPU burst time of LBurst where LBurst is approximately 3 times SBurst.

3Burst = LBurst

- CPU-bound jobs where the time to complete the job varies widely.

One of the general goals of the scheduling algorithm is to minimize overhead by minimizing the number of context switches. A second goal is to prevent starvation.

a) Given the categories above and your understanding of the example CPU scheduling examples we investigated, list one additional scheduling goal our algorithm should satisfy. Explain how this goal will affect both I/O and CPU bound jobs.

1. minimize context switch
2. starvation (fairness)
3. minimize wait (response) time
    + important for interactive (IO bound) jobs
    + idea is to run interactive jobs more than CPU intensive jobs.
    + not so much for CPU-bound jobs

b) Describe how a scheduling algorithm might use priorities to satisfy the above goals.

priority

highest: type 1 —> lowest: type 3

c) Describe the factors to consider when choosing an appropriate time quantum for a pre-emptive scheduler. Is it appropriate to have different sized quanta for different priorities?

since already know length of different jobs
just use different quanta for each priority to
reduce context switch time

q -> SBurst for type 1 jobs
q -> LBurst for type 2 jobs
q -> anything more than LBurst, as long as
possible (avoid context switch time) but also to let
interactive job run

d) Since we have so much information about a job when it arrives, can we assign it a priority and never change it, or do we still need dynamically changing priorities? Explain.

idea is if job is CPU intensive and after running for some time, change to IO operations, at this time, since priority set beforehand (cant set dynamically) so cant change its priority higher such that it gets to run its IO bound operations. This might cause starvation for CPU bound jobs if a lot of interactive job is running at the same time

level 0 -> quantum 2 (preempted)
level 1 -> FCFS scheduling, no quantum limitation (non-preempted)

Question 2:

An OS uses a multi-level feedback scheduler with 2 levels. New and returning processes start at level 0, which uses round robin scheduling with a quantum of 2. A process that uses its entire quantum at level 0 gets moved to level 1, which uses first-come-first-served scheduling. The scheduler always chooses a process from the lowest-numbered non-empty level. Admission of a new process or a process returning from IO does not force a scheduling decision; those processes are placed on the level 0 queue to wait until a scheduling decision is to be made.

Initially, there is one process, P0. New processes P1 and P6 are created at times 1 and 6 respectively. Those are the only three processes you need to schedule. Each process has a CPU burst of 5 time units, an IO burst of 3 time units, and a CPU burst of 1 time unit. During an IO burst, a process is blocked (on a wait queue); it does not require the processor.

Assume context switches take no time. Assume that a new process arrives after the scheduling decision is made for that time slot. Fill in the timeline below for each process. Note when a process is **New**, when it has the **CPU**, when it loses the CPU due to a **Preempt**, what queue it is on (**L0**, **L1**, **IO**), and when it Exits. The first 5 time units have been filled in for you.

P0, P1, P6 has 5 CPU burst, 3 IO burst (on wait queue), and 1 CPU burst

Text

Time	0	1	2	3	4	5	6
P0	CPU	CPU	queue on level 1 Preempt, L1	L1	CPU both on level 1 now	CPU	CPU
P1		New, L0	CPU	CPU	Preempt, L1	L1	L1
P6							New L0
Time	**7**	**8**	**9**	**10**	**11**	**12**	**13**
P0	I/O	I/O	change to interactive I/O	L0	L0	CPU	Exit
P1	L1	L1	CPU	CPU	CPU	I/O	I/O
P6	CPU	CPU	preempt, L1	L1	L1	L1	CPU
Time	**14**	**15**	**16**	**17**	**18**	**19**	**20**
P0							
P1	I/O	L0	CPU	Exit			
P6	CPU	CPU	I/O	I/O	I/O	CPU	Exit

For your solution above, what is the total wait time (time spent ready but not executing) for all processes?

For your solution above, what is the average response time (time spent waiting for the processes's first time slice)?