EE 4770 **Homework 6** **Due: 30 April 1997**

Problem 1: The tasks below are run on an otherwise empty system having a quantum of 10 ms, using first-come, first-served scheduling, and which is not task preemptive.

Task Name	Creation Time / ms	Run Time / ms	Other
A	0	∞	15 until 30
B	7	∞	19 for 8
C	22	21	Nothing Special

Task A computes for 15 ms then sleeps (goes into the wait state); it wakes up (moves to ready) at the next multiple of 30 ms. (That is it's woken up at $t = 30$ ms, $t = 60$ ms, etc.) After waking up it performs another 15 ms of computation and sleeps again, to be woken up at the next multiple of 30 ms.

Task B performs I/O after *every* 19 ms of computation; the I/O takes 8 ms to complete. That is, after each 19 ms of computation B will perform the I/O.

Show the states of the CPU and tasks from $t = 0$ to 100 ms.

Problem 2: The tasks in the table below are run on an otherwise empty system having a quantum of 11 ms and which is not task preemptive.

Task Name	Creation Time / ms	Round 1 Class	Round 2 Deadline	Run Time / ms
A	0	1	N/A	20
B	10	1	N/A	20
C	20	1	N/A	20
D	30	2	60	20
E	40	2	95	20

A multilevel scheduling scheme is used with round robin used in the first round. In the second round first-come, first-served is used for class-1 tasks and deadline scheduling is used for class-2 tasks. Show the states of each task and the which task the CPU is running from $t = 0$ until the last task finishes.

Problem 3:

Task Name	Creation Time / ms	Round 1 Class	Round 2 Deadline	Run Time / ms
A	0	1	N/A	∞
B	10	1	N/A	∞
C	20	1	N/A	∞
D	30	2	60	20
E	40	2	95	20

Like the tasks in the previous problem the class-2 tasks in the table above, which have deadlines, must share CPU with the class-1 tasks. Suppose that tasks A and B must run regularly, but that task C could wait. Show how the scheduling could be modified so that the running of C could not cause D, E, or any new class-2 tasks to miss deadlines, but A and B still get CPU time regularly. (Of course, C must run some time.)

The solution must describe how the scheduling algorithms presented in class can be used.