Problem 1 Part 1 Solution  The output of a simulator showing relevant state changes appears below.

** Time:  0  
Task A created.  
Task A changing from Ready to Run

** Time:  10  
Task B created.

** Time:  20  
Task A requests unavailable resources.  
Task A changing from Run to Wait  
Task B changing from Ready to Run

** Time:  22  
Task C created.  
Task B changing from Run to Ready  
Task C changing from Ready to Run

** Time:  30  
Resources now available for task A.  
Task A changing from Wait to Ready

** Time:  31  
Task C finishes normally.  
Task C changing from Run to Zombie  
Task A changing from Ready to Run

** Time:  44  
Task A finishes normally.  
Task A changing from Run to Zombie  
Task B changing from Ready to Run

** Time:  53  
Task B finishes normally.  
Task B changing from Run to Zombie
Problem 1 Part 2 Solution  The output of a simulator showing relevant state changes appears below.

** Time: 0
Task A created.
Task A changing from Ready to Run

** Time: 10
Task B created.

** Time: 10
Task A quantum expired.
Task A changing from Run to Ready
Task A changing from Ready to Run

** Time: 20
Task A requests unavailable resources.
Task A changing from Run to Wait
Task B changing from Ready to Run

** Time: 22
Task C created.

** Time: 30
Resources now available for task A.
Task A changing from Wait to Ready

** Time: 30
Task B quantum expired.
Task B changing from Run to Ready
Task C changing from Ready to Run

** Time: 39
Task C finishes normally.
Task C changing from Run to Zombie
Task A changing from Ready to Run

** Time: 49
Task A quantum expired.
Task A changing from Run to Ready
Task A changing from Ready to Run

** Time: 52
Task A finishes normally.
Task A changing from Run to Zombie
Task B changing from Ready to Run

** Time: 53
Task B finishes normally.
Task B changing from Run to Zombie
**Problem 2 Solution** Have C arrive before A starts I/O, for example at time 19. Task A will then have to wait for C to finish before making its I/O request only to wait again for the I/O to finish. Such a situation is shown in the simulator output below. An intelligent scheduler would not preempt A when C arrived despite the fact that C's deadline is sooner.

**Time:** 0
Task A created.
Task A changing from Ready to Run

**Time:** 10
Task B created.

**Time:** 19
Task C created.
Task A changing from Run to Ready
Task C changing from Ready to Run

**Time:** 28
Task C finishes normally.
Task C changing from Run to Zombie
Task A changing from Ready to Run

**Time:** 29
Task A requests unavailable resources.
Task A changing from Run to Wait
Task B changing from Ready to Run

**Time:** 39
Resources now available for task A.
Task A changing from Wait to Ready
Task B changing from Run to Ready
Task A changing from Ready to Run

**Time:** 52
Task A finishes normally.
Task A changing from Run to Zombie
Task B changing from Ready to Run

**Time:** 53
Task B finishes normally.
Task B changing from Run to Zombie