**22C:166 Assignment 1: Solution keys**

**Answer 1.** Both are false.

![Diagram](image1)

(i) false  
(ii) false

**Answer 2.** The answer is “any number of times” - it can be arbitrarily large. To see how, consider the case when both flags are true, assume that process 0 pauses for a LONG time after step 5. The other process should be able to get in and out its CS as many times as it wants!

**Answer 3.** For strongly fair schedulers, it is trivial: each philosopher executes:

```plaintext
    do left book AND right book vd use them; return them od
```

(the guard is true infinitely often, so eventually the action will take place)

For a weakly fair scheduler, several solutions are possible. One is to number the forks as 0, 1, 2 and let each philosopher grab one book at a time, always starting with the lower numbered book. Deadlock is impossible, and progress is guaranteed. Note that with three philosophers, it is essentially a mutual exclusion problem. So you can any of the accepted solutions to that problem. (A turn-based solution that encourages round robin scheduling is considered bad, since it causes the fast philosophers to suffer due to the slowness of the others.)

**Answer 4.** A process enters its CS when it receives \( \geq N-L \) acknowledgments. Everything else remains unchanged.

**Answer 5.** Starting from an arbitrary initial configuration, the system of processes settles to a behavior in which the even and odd processes alternately execute their actions. So the maximum number of processes eligible to concurrently execute their actions is the ceiling of \( \lceil n/2 \rceil \)