Eli Sobylak
1-dec-15

Ch 6: 6.1, 6.2, 6.4, 6.5
6.6, 6.11, 6.14, 6.15, 6.16, *6.18

6.1:
Mutual exclusion: This applies to figure 6.1a with the cars because only one car can be using a given section of the road at a given time
Hold and wait: They are all waiting for each other to go first in this situation
No preemption: Once a car starts moving, there's not much you can do to get it out of the way
Circular wait: If a car has already started moving into the turn lane, or if they all make a turn into their turns they will all be deadlocks.

6.2:
Two key factors of deadlock avoidance that could be applied to the scenario in figure 6.1 are not starting a process if its demands might lead to deadlock, and don't grant an incremental resource request to a process if this allocation might lead to deadlock. Obviously there are rules on the road so at a four way stop, this situation doesn't seem to happen, and when it does, someone has to make a decision to go first.

6.4:
Deadlock occurs inevitably in the fatal region. The existence of a fatal region depends on the logic of the two process. These two processes do not run into that problem.

6.15:

c= {3,2,9,7} a= {1,1,3,2}

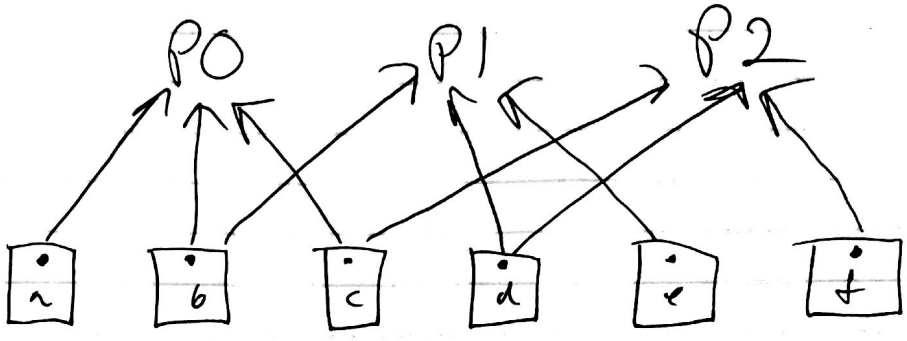
n = {2,1,6,5}

6.16:
a.
1. 3
2. 2
3. 1
4. 6
5. 5
6. 4

OS HW 6

6.6

a.



b.

6.5

Max Claim:
Each process provides a max for all over claim

Need Matrix = Max - Alloc

a) Check Avail
total = alloc + Avail

	Alloc A	B	C	D	Max A	B	C	D	Need A	B	C	D
P0	2	0	2	1	9	5	5	5	7	5	3	4
P1	0	1	1	1	2	2	3	3	2	1	2	2
P2	4	1	0	2	7	5	4	4	3	4	4	2
P3	1	0	0	1	3	3	3	2	2	3	3	1
P4	1	1	0	0	5	2	2	1	4	1	2	1
P5	1	0	1	1	4	4	4	4	3	4	3	3

Avail = A B C D
= 6 3 5 4

Avail A	B	C	D	
6	4	6	5	P1
10	5	6	7	P2
11	5	6	8	P3
12	6	6	8	P4
13	6	7	9	P5
15	6	9	10	P0

6.11

	Curr Alloc A	Max Claim A	Need: Max - Alloc A
P1	45	70	25
P2	40	60	20
P3	15	60	45
			35
a) P4	25	60	~~Avail = 150~~
b) P4	35	60	25

a)

Avail = 150

P1	195
P2	235
P3	250
P4	275

b)

Avail = 150

P1	195
P2	235
P3	250
P4	285