Solutions for CP Exercises, October 6

1. Solution for Silberschatz, Exercise 6.8

Two of the processes may always get their resources at the same time, but not all three of them. One could imagine a deadlock situation in which each process has one instance and waits for another one. But since the three waiting processes only hold three instances, there will be one instance left and thus one of the processes can get its last resource and terminate. Thus, the system cannot enter a deadlock situation.

2. Solution for Silberschatz, Exercise 6.9

Generalizing the argument of exercise 6.8 we get:

For a sytem with m inscances of a resource type, a deadlock situation is characterized by a number of processes that are requesting more instances while holding some already, but no more instances are available.

A process P_i can request more instances only if it has not yet reached is maximal claim MAX_i. The maximal number of instances n processes may have reserved without having reached their maximum claim (and thereby be able to finish) is given by:

$$\sum_{i=1}^{n}(MAX_i - 1) = (\sum_{i=1}^{n} MAX_i) - n = MAX - n$$

Thus, no deadlock can occur if this number is less than the number of available instances m:

$$MAX - n < m$$

or

$$MAX < n + m$$

It is assumed that all processes need several instances, ie. $MAX_i > 1$ for all i and, of course, that $MAX_i \leq m$.

3. Solution for Silberschatz, Exercise 6.15

(a) For each process i, $Need_i = Max_i - Allocation_i$, ie.

	Need			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

(b) The exucution of the *safe* algoritm may be written as follows:

Free (Work)				*Can be finished*
A	B	C	D	
1	5	2	0	P_0
1	5	3	2	P_2
2	8	8	6	P_1
3	8	8	6	P_3
3	14	11	8	P_4
3	14	12	12	

In each step, a process to be finished is found by comparing *Free* with *Need*. Since the algoritm terminates with all processes finished, the situation is *safe*.

(c) First, the sitiation is changed as if the request had been granted:

$$\begin{aligned} Allocation_1 &= (1,4,2,0) \\ Need_1 &= (0,3,3,0) \\ Free &= (1,1,0,0) \end{aligned}$$

Then the *safe* algoritm is performed again:

Free (Work)				*Can be finished*
A	B	C	D	
1	1	0	0	P_0
1	1	1	2	P_2
2	4	6	6	P_1
3	8	8	6	P_3
3	14	11	8	P_4
3	14	12	12	

Since the request $(0,4,2,0)$ from P_1 would result in a situation that is still safe, the request can be granted.