Operating System HW4

B04901096 蔡昕宇

 # 7.1

(a) The four necessary conditions are: **mutual exclusion**, **hold-and-wait**, **circular wait** and **no preemption**.
The mutual exclusion condition holds since only one car can occupy a space in the road at a time.
Hold-and-wait condition holds where some cars are holding on a road and willing to occupy the another.
Circular wait holds since each cat is waiting the another car to move on. We can easily trace from the first car to the end one and then back to the first one.
No preemption is hold since no any car can be removed from its place.

(b) Simple rule is that each car is not able to stay at the intersection until the condition in the front of the car is clear.

 # 7.8

From the two conditions (a) and (b), we have

$$1 \leq Max_i \leq m, \ \forall i$$

$$\sum_{i=0}^{n} Max_i < m + n$$

If there exists a deadlock state, all resources should be allocated. Therefore,

$$\sum_{i=0}^{n} Allocation_i = m$$

And also we know that

$$\sum_{i=0}^{n} Need_i + \sum_{i=0}^{n} Allocation_i = \sum_{i=0}^{n} Max_i$$

always holds. Hence, from the condition (b) and the deadlock condition

$$\sum_{i=0}^{n} Need_i + \sum_{i=0}^{n} Allocation_i = \sum_{i=0}^{n} Need_i + m = \sum_{i=0}^{n} Max_i < m + n.$$

Then we can get from the reduction of the equation above

$$\sum_{i=0}^{n} Need_i < n$$

It implies the need of some process P_i equals to 0. Since $Max_i \geq 1$ for process P_i from condition (a), it means that at least one resource that P_i can release. Hence, there should not be a deadlock state and it is deadlock free.

7.13

	Allocation					Max					Need					Available			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P0	2	0	0	1		4	2	1	2		2	2	1	1		3	3	2	1
P1	3	1	2	1		5	2	5	2		2	1	3	1					
P2	2	1	0	3		2	3	1	6		0	2	1	3					
P3	1	3	1	2		1	4	2	4		0	1	1	2					
P4	1	4	3	2		3	6	6	5		2	2	3	3					

(a)
First we check that only the need of P_0 is less than available. Hence, P_0 can be fulfilled and release its resources. The next state will be

	Allocation					Max					Need					Available			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P0	**Done**															5	3	2	2
P1	3	1	2	1		5	2	5	2		2	1	3	1					
P2	2	1	0	3		2	3	1	6		0	2	1	3					
P3	1	3	1	2		1	4	2	4		0	1	1	2					
P4	1	4	3	2		3	6	6	5		2	2	3	3					

We can observe that P_3 can be done at this point. Thus, the allocated resources can be release after P_3 is done.
After P_3 is done, the need of the remaining processes, either P_1, P_2 or P_4, is satisfied. Hence, the system is in a safe state and an example order of executing processes is <P_0, P_3, P_1, P_2, P_4>
(b)
P_1 can request (1, 1, 0, 0) since the available resources is enough for it. The next state will be

	Allocation					Max					Need					Available			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P0	2	0	0	1		4	2	1	2		2	2	1	1		2	2	2	1
P1	4	2	2	1		5	2	5	2		1	0	3	1					
P2	2	1	0	3		2	3	1	6		0	2	1	3					
P3	1	3	1	2		1	4	2	4		0	1	1	2					
P4	1	4	3	2		3	6	6	5		2	2	3	3					

The system is able to complete P_0 at this point and release its resources. A sequence < P_0, P_3, P_1, P_2, P_4>, identical to (a), shows that all process can be done in this order.

(c)
P_4 can request (0, 0, 2, 0) since the available resources is enough for it. The next state will be

	Allocation					**Max**					**Need**					**Available**			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P0	2	0	0	1		4	2	1	2		2	2	1	1		3	3	0	1
P1	3	1	2	1		5	2	5	2		2	1	3	1					
P2	2	1	0	3		2	3	1	6		0	2	1	3					
P3	1	3	1	2		1	4	2	4		0	1	1	2					
P4	1	4	5	2		3	6	6	5		2	2	1	3					

The system is not able to complete any process and a deadlock condition occurs.
Hence, the request from P_4 should not be granted immediately to prevent deadlock.