CS471: Operating System Concepts
Fall 2007
Sample questions/solutions for Chapter 7

Question 1 [Exercise 7.2

The four necessary conditions for a deadlock to occur are:

a) Mutual Exclusion
b) Hold and Wait
c) No Preemption
d) Circular Wait

For Dining Philosophers problem:

a) Mutual exclusion: The five philosophers are trying to use the chopstick at the same time. Since only five chopsticks are there and each philosopher needs two chopsticks to complete eating, this can lead to deadlock.
b) Hold and Wait: Each philosopher is holding one chopstick and requires the other to complete. This satisfies the condition Hold and wait.
c) No preemption: Chopsticks that are held by each philosopher, are not available to other philosophers since they don't give up chopsticks until they ate.
d) Circular wait: Philosopher 1 is waiting for chopstick from philosopher 2 and philosopher 2 is waiting for chopstick from philosopher 2 and so on. This satisfies the Circular wait condition for deadlock.

Example: Avoid Hold and Wait condition by requiring a philosopher to release a chopstick when he cannot acquire the other, we can avoid deadlock situation.

Question 2 Exercise 7.5

a) Increase ***Available*** (new resources added).

This addition always leads to safe state.

b) Decrease ***Available*** **(resource permanently removed from the system)**

Only safe if the banker's algorithm tells us that the system is in a safe state with the reduced number of resources.

c) Increase ***Max*** for one number of process(the process needs more resources than allowed, it may want more)

Only safe if the system will be in a safe state with the increase value of ***Max.***

d) Decrease the ***Max*** for one process (the process decides it does not need that many resources).

Always safe.

e) Increase the number of processes.

Always safe – the number of processes has nothing to do with deadlock. If we have more processes and fewer resources, it will not lead to deadlock. The only time that it might happen is that if a PCB involve in the circular wait. But since the PCB is the first resource that a process gets, it cannot get anything else until it has a PCB. Therefore, no circular wait involve with the PCB.

f) Decrease the number of processes.

Always safe (as mentioned in e)

Question 3 Exercise 7.6

Consider a system consisting of five resources of the same type that are shared by four processes, each of which needs at most two resources. Show that the system is deadlock free.

Solution:

If all four processes need two resources, at least one process will get two resources (Since there are five resources available), while the others get one resource.
The other three processes will have to wait until the one with two resources finishes its job and then release the resources eventually. Therefore, there will be no deadlock in this situation.

Question 4 Exercise 7.11

Consider the following snapshot of a system:

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

Answer the following questions using the banker's algorithm:

a) What is the content of the need matrix ***Need*****?**

The content of matrix *Need* is defined as *Max – Allocation.*

	Need			
	A	**B**	**C**	**D**
P0	**0**	**4**	**3**	**1**
P1	**2**	**3**	**2**	**1**
P2	**1**	**1**	**1**	**1**
P3	**0**	**2**	**0**	**1**
P4	**1**	**1**	**1**	**2**

b) Is the system in a safe state?

The available number of resources of A B C D are 1 1 1 1 respectively.

Since process P2 need 1 1 1 1 of resources type A B C D respectively, the available resource 1 1 1 1 can be allocated to process P2.

After P2 finishes, available resources are 2 2 2 2.

The available resource 2 2 2 2 can satisfy the needs of either process P3 or P4. Let process P3 be given the resources that are needed.

After P3 finishes, the available resources are 2 2 4 5.

The available resources 2 2 4 5 can satisfy the needs of the process P4.

Let process P4 given the resources that are needed.

After P4 finishes, the available resources are 3 4 7 9.

The available resources 3 4 7 9 can satisfy the needs of either process P0 or P1.

Let process P1 be given the resources that are needed.

After P1 finishes, the available resources are 5 4 7 13.

The available resources 5 4 7 13 can satisfy the needs of the process P0.

Let process P0 be given the resources that are needed.

After P0 finishes the available resources are 5 6 9 16

Since all the processes can be satisfied, the system is in a safe state.

The sequence < P2, P3, P4, P1, P0 > satisfies the safety criteria.

C) If a request from process P1 arrives for (1, 1, 1, 1) can the request be satisfied immediately?

If process P1 request is satisfied then the available matrix will be 0 0 0 0 for resources A B C D respectively. At this point we don't have any available resources and all the process are holding some resources and still require some more resources to complete.

Therefore process P1 request can't be granted immediately. Doing so will end up in deadlock where all processes hold some resources and need more resources to finish.

Question 5 Exercise 7.14 (Hint: bridge is the resource here that needs to be accessed in a mutually exclusive fashion. There are two processes contending to access it.)

```
semaphore bridge = 1;

Void entering_bridge (direction x)
{
  wait (bridge);
 entering  bridge
 }

Void exiting_bridge(direction x);
{
      Leaving bridge
      signal(bridge);

}
```

Note: This solution assumes that exiting_bridge is only used after entering_bridge. If we want to take care of a case where a spurious process executes exiting_bridge while not having executed entering_Bridge, we need to introduce additional variable such as which direction traffic is currently flowing on the bridge.