COMP 2432 Operating Systems
Tutorial 11

1. Safety of Resource Allocation State.

Given the following two resource allocation states (left and right) for five processes with three types of resources, the number of currently available resources is (3 0 2). What is the total number of each type of resources? Determine whether each of these two states (*a*) and (*b*) is *safe*. Give a *safe sequence* in case that it is safe.

(*a*)

	Allocation			Max		
	A	*B*	*C*	*A*	*B*	*C*
P_0	2	1	0	5	3	3
P_1	1	2	0	3	2	2
P_2	2	0	1	6	3	5
P_3	2	0	1	2	2	2
P_4	0	0	2	2	3	6

(*b*)

	Allocation			Max		
	A	*B*	*C*	*A*	*B*	*C*
P_0	2	2	0	3	2	2
P_1	2	0	0	3	2	3
P_2	1	0	2	5	1	2
P_3	2	1	1	2	2	2
P_4	1	0	2	2	3	3

Suppose that P_4 in (*a*) now refines its maximum demand from (2 3 6) down to (2 3 **5**). Determine whether the same resource allocation state is *safe*.

Suppose that P_3 in (*b*) now refines its maximum demand from (2 2 2) up to (2 **3 4**). Determine whether the same resource allocation state is *safe*.

2. Deadlock Avoidance.

Could the following *requests* by P_3 and P_4 be *granted* with respect to the two resource allocation states (*a*) and (*b*) below? If a request could be granted, what would be an appropriate *safe sequence* for the new state? How many *possible safe sequences* are there for each of the two cases?

(*a*)

	Allocation			Max		
	A	*B*	*C*	*A*	*B*	*C*
P_0	2	1	0	5	3	3
P_1	1	0	0	3	1	2
P_2	2	0	2	4	3	2
P_3	2	1	1	4	1	2
P_4	0	1	2	2	2	3
Avail	3	0	5			
Req_3	2	0	1			

(*b*)

	Allocation			Max		
	A	*B*	*C*	*A*	*B*	*C*
P_0	2	1	0	6	2	3
P_1	3	1	0	3	1	2
P_2	2	1	1	3	1	5
P_3	1	0	2	2	2	2
P_4	1	0	2	4	3	6
Avail	3	0	5			
Req_4	3	0	2			

Suppose that the requesting process in (*a*) is not P_3, but by another unknown process P_x ($x \neq 3$). Without knowing what x is, could the request be *granted*? Would the decision *change* if the requesting process in (*b*) is P_0 instead of P_4?

3. Deadlock Detection.

Determine whether each of the following two resource allocation states represents a *deadlocked state*. Give a *completion sequence* if there is no deadlock. Otherwise, indicate the set of processes *involved in the deadlock*. How many *possible completion sequences* are there for each case?

(*a*)

	Allocation			Request		
	A	*B*	*C*	*A*	*B*	*C*
P_0	2	1	0	0	1	2
P_1	1	0	0	3	0	5
P_2	2	0	2	2	1	1
P_3	2	1	1	0	0	0
P_4	0	1	2	2	0	1
Avail	1	0	0			

(*b*)

	Allocation			Request		
	A	*B*	*C*	*A*	*B*	*C*
P_0	2	1	0	0	1	2
P_1	2	0	1	3	0	4
P_2	0	1	2	2	1	1
P_3	1	0	0	0	0	0
P_4	2	1	1	2	0	2
Avail	1	0	1			

Suppose that the only available resource, *A*, in (*a*) is now **broken**. Is the new resource allocation state *deadlocked*? Consider the resource allocation state in (*b*), in which an instance of resource *B* has been **repaired** and returned to the system, such that the available array becomes (1 **1** 1). Is the state in a *deadlocked state*? If not, give some possible completion sequences.