COMP 2432 Operating Systems
Tutorial 11 Solution

1. Safety of Resource Allocation State.

We first compute the *Need* array for each process on the resources.
For (*a*), ***total*** number of resources = (**10 3 6**).

(*a*)

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

The state is an ***unsafe state***.
Start from *Work* = (3 0 2).
P_1 can finish, *Work* = (3 0 2) + (1 2 0) = (4 2 2). P_3 can finish, *Work* = (4 2 2) + (2 0 1) = (6 2 3).
P_0 can finish, *Work* = (8 3 3). Only P_0, P_1 and P_3 could finish and it is ***not*** *a safe state*.

For (*b*), ***total*** number of resources = (**11 3 7**).

(*b*)

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

The state is a ***safe state***.
A possible safe sequence is $<P_0, P_2, P_1, P_3, P_4>$.
Start from *Work* = (3 0 2).
P_0 can finish, *Work* = (3 0 2) + (2 2 0) = (5 2 2). P_2 can finish, *Work* = (5 2 2) + (1 0 2) = (6 2 4).
P_1 can finish, *Work* = (8 2 4). P_3 can finish, *Work* = (10 3 5).
P_4 can finish, *Work* = (**11 3 7**). So all processes could finish and it is a *safe state*.
Now *Work* = total number of resources (11 3 7) initially (as a checking step).
Other safe sequences include $<P_0, P_2, P_3, P_1, P_4>$, $<P_0, P_2, P_3, P_4, P_1>$, $<P_0, P_3, P_1, P_2, P_4>$, $<P_0, P_3, P_1, P_4, P_2>$, $<P_0, P_3, P_2, P_1, P_4>$, $<P_0, P_3, P_2, P_4, P_1>$, $<P_0, P_3, P_4, P_1, P_2>$, $<P_0, P_3, P_4, P_2, P_1>$, a total of 9 safe sequences.

With the revised maximum demand for (*a*), ***total*** number of resources remains to be (**10 3 6**).

(*a*)(*ii*)

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

Now, the state becomes a ***safe state***. A possible safe sequence is $<P_1, P_3, P_0, P_4, P_2>$.
Start from *Work* = (3 0 2).
P_1 can finish, *Work* = (3 0 2) + (1 2 0) = (4 2 2). P_3 can finish, *Work* = (4 2 2) + (2 0 1) = (6 2 3).
P_0 can finish, *Work* = (8 3 3). P_4 can finish, *Work* = (8 3 5).
P_2 can finish, *Work* = (**10 3 6**). So all processes could finish and it is a *safe state*. Note that this is the ***unique*** safe sequence in this case.

With the revised maximum demand for (*b*), ***total*** number of resources remains to be (**11 3 7**).

(*b*)(*ii*)

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

The state is still a ***safe state***. A possible safe sequence is <P_0, P_2, P_1, P_3, P_4>, same as before, with similar steps above. There are two other safe sequences <P_0, P_2, P_3, P_1, P_4> and <P_0, P_2, P_3, P_4, P_1>, a total of only 3. Since the demand is increased, it can be anticipated that there would possibly be fewer ways to complete the processes, i.e. possibly fewer safe sequences.

2. Deadlock Avoidance.

For (*a*), total number of resources = (**10 3 10**). There are sufficient resources for the request. Pretend that the request by P_3 is granted. *Avail* = (1 0 4).

(*a*)

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

The resultant state is a ***safe state***. A possible safe sequence is <P_3, P_1, P_4, P_0, P_2>.
Start from *Work* = (1 0 4).
P_3 can finish, *Work* = (1 0 4) + (4 1 2) = (5 1 6). P_1 can finish, *Work* = (5 1 6) + (1 0 0) = (6 1 6).
P_4 can finish, *Work* = (6 2 8). P_0 can finish, *Work* = (8 3 8).
P_2 can finish, *Work* = (**10 3 10**). So all processes could finish. Thus, the request can be ***granted***.
There are three more possible safe sequences: <P_3, P_4, P_0, P_1, P_2>, <P_3, P_4, P_0, P_2, P_1>, and <P_3, P_4, P_1, P_0, P_2>, with a total of 4.

For (*b*), total number of resources = (**12 3 10**). There are sufficient resources for the request. Pretend that the request by P_4 is granted. *Avail* = (0 0 3).

(*b*)

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

Start from *Work* = (0 0 3). P_1 can finish and *Work* = (0 0 3) + (3 1 0) = (3 1 3). Now, no other process could finish in the worst scenario as *Work* is not sufficient. The resultant state is an ***unsafe state***, since only P_1 could finish. Thus, the request ***cannot*** be granted.
There is *no* safe sequence, i.e., the count is ***zero***.

When the requesting process in (*a*) is not P_3, but P_x where $x \neq 3$. the request ***cannot*** be granted. This is because pretending that Req_x has been granted, regardless of the value of *x*, all the 4 resultant states are ***unsafe***. You should proceed to verify this statement by repeating the steps in (*a*). In fact, if the request is raised by P_2, the request should be directly *rejected*, since the total request on resource *C* exceeds the maximum demand declared by P_2.

When the requesting process in (*b*) is P_0 instead of P_4, the request could be granted. Total number of resources = (**12 3 10**). There are sufficient resources for the request by P_0. Pretend that the request is granted. *Avail* = (0 0 3).

(*b*)(*ii*)

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

The resultant state after the pretended resource allocation remains ***safe***.
A possible safe sequence is <P_1, P_0, P_2, P_3, P_4>.
Start from *Work* = (0 0 3).
P_1 can finish, *Work* = (3 1 3). P_0 can finish, *Work* = (8 2 5).
P_2 can finish, *Work* = (10 3 6). P_3 can finish, *Work* = (11 3 8).
P_4 can finish, *Work* = (**12 3 10**). So all processes could finish. Thus, the request can be ***granted***.
There are two more possible safe sequences: <P_1, P_0, P_2, P_4, P_3> and <P_1, P_0, P_3, P_2, P_4>, with a total of 3.

3. Deadlock Detection.

For (*a*), total number of resources = (**8 3 5**). The state is ***not*** a deadlocked state. A possible completion sequence is <P_3, P_2, P_0, P_4, P_1>.
Start from *Work* = (1 0 0).
P_3 can finish, *Work* = (1 0 0) + (2 1 1) = (3 1 1). P_2 can finish, *Work* = (3 1 1) + (2 0 2) = (5 1 3).
P_0 can finish, *Work* = (7 2 3). P_4 can finish, *Work* = (7 3 5).
P_1 can finish, *Work* = (**8 3 5**). So all processes could finish successfully and there is no deadlock.
There are five more possible completion sequences: <P_3, P_2, P_4, P_0, P_1>, <P_3, P_2, P_4, P_1, P_0>, <P_3, P_4, P_0, P_2, P_1>, <P_3, P_4, P_2, P_0, P_1>, and <P_3, P_4, P_2, P_1, P_0>, with a total of 6.

For (*b*), total number of resources = (**8 3 5**). The state is ***deadlocked***.
Starting from *Work* = (1 0 1). P_3 can finish and *Work* = (1 0 1) + (1 0 0) = (2 0 1).
Now, no other process can finish.
Thus, only P_3 can finish and all other processes, i.e., P_0, P_1, P_2, and P_4 are involved in the deadlock.
There is *no* completion sequence, i.e., the count is ***zero***.

If the available resource *A* is now **broken** in (*a*), total number of resources = (**7 3 5**). Luckily, the state is still ***not*** a deadlocked state. A possible completion sequence is <P_3, P_2, P_0, P_4, P_1>.
Start from *Work* = (0 0 0).
P_3 can finish, *Work* = (0 0 0) + (2 1 1) = (2 1 1). P_2 can finish, *Work* = (2 1 1) + (2 0 2) = (4 1 3).
P_0 can finish, *Work* = (6 2 3). P_4 can finish, *Work* = (6 3 5).
P_1 can finish, *Work* = (**7 3 5**). So all processes could finish successfully and there is *no deadlock*.
There are five more possible completion sequences: <P_3, P_2, P_4, P_0, P_1>, <P_3, P_2, P_4, P_1, P_0>, <P_3, P_4, P_0, P_2, P_1>, <P_3, P_4, P_2, P_0, P_1>, and <P_3, P_4, P_2, P_1, P_0>, with a total of 6.

If an instance of resource *B* is now **repaired** in (*b*), total number of resources = (**8 4 5**). The state is ***not*** a deadlocked state. A possible completion sequence is <P_3, P_2, P_0, P_4, P_1>.
Start from *Work* = (1 1 1).
P_3 can finish, *Work* = (1 1 1) + (1 0 0) = (2 1 1). P_2 can finish, *Work* = (2 1 1) + (0 1 2) = (2 2 3).
P_0 can finish, *Work* = (4 3 3). P_4 can finish, *Work* = (6 4 4).
P_1 can finish, *Work* = (**8 4 5**). So all processes could finish successfully and there is *no deadlock*.
There are two more possible completion sequences: <P_3, P_2, P_4, P_0, P_1> and <P_3, P_2, P_4, P_1, P_0>, with a total of 3.