CSI3131 – Operating Systems
Tutorial 6
Deadlocks – Solution

1. Is it possible to have a deadlock involving only one single process? Justify your answer.

   No. This follows directly from the hold-and-wait condition, that is, with a single process, it is impossible that the hold and wait condition exists.

2. Consider a computer system that runs 5,000 jobs per month with no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about 10 jobs per deadlock. Each job is worth about $2 (in CPU time), and the jobs terminated tend to be about half-done when they are aborted.

   A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker's algorithm) could be installed in the system with an increase in the average execution time per job of about 10 percent. Since the machine currently has 30-percent idle time, all 5,000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average.
   - What are the arguments for installing the deadlock-avoidance algorithm?
   - What are the arguments against installing the deadlock-avoidance algorithm?

   An argument for installing deadlock avoidance in the system is that we could ensure deadlock would never occur. In addition, despite the increase in turnaround time, all 5,000 jobs could still run. An argument against installing deadlock avoidance software is that deadlocks occur infrequently and they cost little when they do occur. The time used for executing deadlock could be used to run additional jobs.

3. Recall the following example in class:
   - 5 processes are running in the system; *P*0 through *P*4.
   - They are using 3 resource types *A* (with 10 instances), *B* (with 5 instances), and *C* (with 7 instances).
   - At time T0, the snapshot of data structures maintained by the OS are as follows:

Process	Allocation Matrix	Max Matrix	Need Matrix	Available Vector
	A B C	A B C	A B C	A B C
*P*0	0 1 0	7 5 3	7 4 3	3 3 2
P1	2 0 0	3 2 2	1 2 2	
P2	3 0 2	9 0 2	6 0 0	
P3	2 1 1	2 2 2	0 1 1	
P4	0 0 2	4 3 3	4 3 1	

(a) Can request for (3,3,0) by P4 be granted?

Request (3, 3, 0) < Available (3, 3, 2), then continue;
Request (3, 3, 0) < Need[4] (4, 3, 1), then continue
Update the data structures as if the request is granted

Process	Allocation Matrix	Max Matrix	Need Matrix	Available Vector
	A B C	A B C	A B C	A B C
P0	**0 1 0**	**7 5 3**	**7 4 3**	**0 0 2**
P1	**2 0 0**	**3 2 2**	**1 2 2**	
P2	**3 0 2**	**9 0 2**	**6 0 0**	
P3	**2 1 1**	**2 2 2**	**0 1 1**	
P4	**3 3 2**	**4 3 3**	**1 0 1**	

Apply the safety algorithm:

Set Work (0, 0, 2) and Finish (F, F, F, F, F)
Search for a process that can terminate,
No process can have its needs (Need Matrix) met by the available resources (Available vector), that is, no Need[i] ≤ Work.
Terminate search.
Since Finish contains processes that have not terminated, the above state is unsafe and the request by P4 for the resources is not granted – P4 must wait.

(b) Can request for (0,2,0) by P0 be granted?

Request (0, 2, 0) < Available (3, 3, 2), then continue;
Request (0, 2, 0) < Need[0] (7, 4, 3), then continue
Update the data structures as if the request is granted

Process	Allocation Matrix	Max Matrix	Need Matrix	Available Vector
	A B C	A B C	A B C	A B C
P0	0 3 0	7 5 3	7 2 3	3 1 2
P1	2 0 0	3 2 2	1 2 2	
P2	3 0 2	9 0 2	6 0 0	
P3	2 1 1	2 2 2	0 1 1	
P4	0 0 2	4 3 3	4 3 1	

Apply the safety algorithm:

Set Work (3, 1, 2) and Finish (F, F, F, F, F)
Search for a process that can terminate,
P3 can terminate, Need[3] (0, 1, 1) ≤ Work (3, 1, 2)
Update Work to Work+Allocation[3]: (5, 2, 3), Set Finish[3] to true: Finish (F, F, F, T, F)
Search for a process that can terminate,
P1 can terminate, Need[1] (1, 2, 2) ≤ Work (5, 2, 3)
Update Work to Work+Allocation[1]: (7, 2, 3), Set Finish[1] to true: Finish (F, T, F, T, F)
Search for a process that can terminate,
P0 can terminate, Need[0] (7, 2, 3) ≤ Work (7, 2, 3)
Update Work to Work+Allocation[0]: (7, 5, 3), Set Finish[0] to true: Finish (T, T, F, T, F)
Search for a process that can terminate,
P2 can terminate, Need[2] (6, 0, 0) ≤ Work (7, 5, 3)
Update Work to Work+Allocation[2]: (10, 5, 5), Set Finish[2] to true: Finish (T, T, T, T, F)
Search for a process that can terminate,
P4 can terminate, Need[4] (4, 3, 1) ≤ Work (10, 5, 5)
Set Finish[4] to true: Finish (T, T, T, T, T)

Terminate Search.
Since all processes are finished, the above state is safe and the request by P0 for the resources is granted.

4. Given the following system state that defines how 4 types of resources are allocated to 5 running processes.

Available = {2, 1, 0, 0 }

$$\textbf{Allocation} = \begin{array}{c} \\ P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{array} \begin{array}{c} \begin{array}{cccc} r0 & r1 & r2 & r3 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 2 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 \\ 2 & 3 & 5 & 4 \\ 0 & 3 & 3 & 2 \end{pmatrix} \end{array} \quad \textbf{Max} = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 2 & 7 & 5 & 0 \\ 6 & 6 & 5 & 6 \\ 4 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \end{pmatrix}$$

a) Complete the Need matrix :

$$\textbf{Need} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 6 & 6 & 2 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 3 & 2 & 0 \end{pmatrix}$$

b) Is the system in a safe state? Why or why not? If in a safe state, give a safe sequence.

Apply the safety algorithm to see if in a safe state:
Initialize: Work = {2, 1, 0, 0 } Finish = {F, F, F, F,F}
Select P0 that does not need any additional resources
Work = {2,1,0,0}+{0,0,1,2}={2, 1, 1, 2 }
Finish = {T,F,F,F,F}
Select P3 that can request up to {2,0,0,2}
Work = {2,1,1,2}+{2,3,5,4} = {4,4,6,6}
Finish= {T,F,F,T,F}
Select P4 that can request up to {0,3,2,0}
Work = {4,4,6,6}+{0,3,3,2} = {4,7,9,8}
Finish= {T,F,F,T,T}
Select P1 that can request up to {0,7,5,0}
Work = {4,7,9,8}+{2,0,0,0} = {6,7,9,8}
Finish= {T,T,F,T,T}
Select P2 that can request up to {6,6,2,2}
Work = {6,7,9,8}+{0,0,3,4} = {6,7,12,12}
Finish= {T,T,T,T,T}
A safe sequence was found : <P0,P3,P4,P1,P2>

c) For each of the following requests:
   a. P0: *Request* = {0, 1, 0, 0}
   b. P1: *Request* = {0, 1, 0, 0}
   c. P2: *Request* = {0, 1, 0, 0}
   d. P3: *Request* = {0, 0, 0, 1}

Determine if the request should be granted. If it can be granted, show that the system is in a safe and give a safe sequence.

P0: *Request* = {0, 1, 0, 0}
Cannot grant request, P0 has been allocated maximum resources, i.e. *Request[i] ≤ Need[0,i] for all i* is not TRUE. An error is returned to P0.

P1: *Request* = {0,1,0,0}
Pretend to grant the request and update the allocation state:

Available = {2, 0, 0, 0 }

$$\textbf{Allocation} = \begin{array}{c} \\ P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{array} \begin{array}{c} \begin{array}{cccc} r0 & r1 & r2 & r3 \end{array} \\ \begin{Bmatrix} 0 & 0 & 1 & 2 \\ 2 & 1 & 0 & 0 \\ 0 & 0 & 3 & 4 \\ 2 & 3 & 5 & 4 \\ 0 & 3 & 3 & 2 \end{Bmatrix} \end{array} \qquad \textbf{Max} = \begin{Bmatrix} 0 & 0 & 1 & 2 \\ 2 & 7 & 5 & 0 \\ 6 & 6 & 5 & 6 \\ 4 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \end{Bmatrix}$$

$$\textbf{Need} = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6 & 5 & 0 \\ 6 & 6 & 2 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 3 & 2 & 0 \end{Bmatrix}$$

Is projected state a safe state?

Initialize: **Work = Available = {2, 0, 0, 0 }**
Finish = {F, F, F, F, F}
Select P0 that does not need any additional resources
Work = {2,0,0,0}+{0,0,1,2}={2, 0, 1, 2 }
Finish = {T,F,F,F,F}
Select P3 that can request up to {2,0,0,2}
Work = {2, 0, 1, 2}+{2,3,5,4} = {4,3,6,6}
Finish= {T,F,F,T,F}
Select P4 that can request up to {0,3,2,0}
Work = {4,3,6,6}+{0,3,3,2} = {4,6,9,8}
Finish= {T,F,F,T,T}
Select P1 that can request up to {0,6,5,0}
Work = {4,6,9,8}+{2,1,0,0} = {6,7,9,8}
Finish= {T,T,F,T,T}
Select P2 that can request up to {6,6,2,2}
Work = {6,7,9,8}+{0,0,3,4} = {6,7,12,12}
Finish= {T,T,T,T,T}

A safe sequence was found : <P0,P3,P4,P1,P2>

P2: *Request* = {0, 1, 0, 0}
Pretend to update the request and update the matrices:

Available = {2, 0, 0, 0 }

$$\textbf{Allocation} = \begin{array}{c} \\ P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{array} \begin{array}{c} \begin{array}{cccc} r0 & r1 & r2 & r3 \end{array} \\ \begin{Bmatrix} 0 & 0 & 1 & 2 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 3 & 4 \\ 2 & 3 & 5 & 4 \\ 0 & 3 & 3 & 2 \end{Bmatrix} \end{array} \qquad \textbf{Max} = \begin{Bmatrix} 0 & 0 & 1 & 2 \\ 2 & 7 & 5 & 0 \\ 6 & 6 & 5 & 6 \\ 4 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \end{Bmatrix}$$

Complete the Need matrix :

$$\textbf{Need} = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 6 & 5 & 2 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 3 & 2 & 0 \end{Bmatrix}$$

Is projected state a safe state?

Initialize: **Work = Available = {2, 0, 0, 0 }**
Finish = {F, F, F, F, F}
Select P0 that does not need any additional resources
Work = {2,0,0,0}+{0,0,1,2}={2, 0, 1, 2 }
Finish = {T,F,F,F,F}
Select P3 that can request up to {2,0,0,2}
Work = {2, 0, 1, 2}+{2,3,5,4} = {4,3,6,6}
Finish= {T,F,F,T,F}
Select P4 that can request up to {0,3,2,0}
Work = {4,3,6,6}+{0,3,3,2} = {4,6,9,8}
Finish= {T,F,F,T,T}

Neither P1 nor P2 can be satisfied with the resources available in the Work vector. Thus no safe sequence can be found, and state is not safe – refuse the request.

P3: *Request* = {0, 0, 0, 1}
Cannot grant request, P3 requesting more resources than available, i.e. *Request[i]* <= *Available[i] for all i* is not true. P3 must wait.

5. Deadlock Detection

Given the following system resource allocation statete:

Available = { 2, 1, 0, 0 }

$$\textbf{Allocation} = \begin{Bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{Bmatrix} \qquad \textbf{Request} = \begin{matrix} P0 \\ P1 \\ P2 \end{matrix} \begin{Bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{Bmatrix}$$

a) Determine if a deadlock exists.

Initialize:
Work = {2,1,0,0}
Finish = {F,F,F}

Select P2 for termination, Work contains enough resources to satisfy P2.
Work = {2,1,0,0} + {0,1,2,0} = {2,2,2,0}
Finish = {F,F,T}
Select P1 for termination, Work contains enough resources to satisfy P1.
Work = {2,2,2,0} + {2,0,0,1} = {4,2,2,1}
Finish = {T,F,T}
Select P0 for termination, Work contains enough resources to satisfy P0.
Work = {4,2,2,1} + {0,0,1,0} = {4,2,3,1}
Finish = {T,T,T}

A safe sequence has been found <P2, P1, P0>, no deadlock exists.

b) Illustrate the system with a resource allocation graph.