Instructor: Jungchan Cho (thinkai@gachon.ac.kr),

Deadlock Detection – Several Instance of a Resource Type

The wait-for graph scheme is not applicable to a resource-allocation system with multiple instances of each resource type. We need a deadlock detection algorithm that is applicable to such a system. The algorithm employs several time-varying data structure that are similar to those used in the banker's algorithm.

- **Available**. A vector of length m indicates the number of available resources of each type.
- **Allocation**: An $n \times m$ matrix defines the number of resources of each type currently allocated to each thread
- **Request**: An $n \times m$ matrix indicates the current request of each thread. If Request[i][j] equals k, then thread T_i is requesting k more instances of resource type R_j

The $\leq$ relation between two vectors denotes as follows:

Let X and Y be vectors of length n. We say that $X \leq Y$ if and only if $X[i] \leq Y[i]$ for all $i = 1,2,..,n$. For example, if $X = (1,7,3,2)$ and $Y = (0,3,2,1)$, then $Y \leq X$. In addition, $Y < X$ if $Y \leq X$ and $Y \neq X$.

We can treat each row in the matrices $\boldsymbol{Allocation}$ and $\boldsymbol{Request}$ as vectors and refer to them as $\boldsymbol{Allocatoin_i}$ and $\boldsymbol{Request_i}$. The vector $\boldsymbol{Allocation_i}$ specifies the resources currently allocated to thread T_i; the vector $\boldsymbol{Request_i}$ specifies the resources requested by thread T_i.

This detection algorithm simply investigates every possible allocation sequence for the threads that remain to be completed.

1. Let *Work* and *Finish* be vectors of length m and n, respectively. Initialize $Work = Available$. For $i = 0, 1, ..., n-1$, if $Allocation_i \neq 0$, then $Finish[i] = false$. Otherwise, $Finish[i] = true$.
2. Find an index i such that both
   a. $Finish[i] == false$
   b. $Request_i \leq Work$

   If no such i exists, go to step 4.
3. $Work = Work + Allocation_i$
   $Finish[i] = true$
   Go to step 2.
4. If $Finish[i] == false$ for some i, $0 \leq i < n$, then the system is in a deadlocked state. Moreover, if $Finish[i] == false$, then thread T_i is deadlocked.

[Problem]

Consider a system with five threads T_0 through T_4 and three resource types A, B, and C. Resource type A has seven instances, resource type B has two instances, and resource type C has six instances.

	Allocation	***Request***	***Available***
	A B C	A B C	A B C
T_0	0 1 0	0 0 0	0 0 0
T_1	2 0 0	2 0 2	
T_2	3 0 3	0 0 0	
T_3	2 1 1	1 0 0	
T_4	0 0 2	0 0 2	

1) Answer whether the system below is in the deadlock state. If there does not exist a deadlock state, solve the problem using the algorithm presented above and write down the process in detail and find the sequence in which the system works without a deadlock. If there exists a deadlock, write down all the threads, consisting of the deadlock.

2) Suppose now that T_2 makes one additional request for an instance of type C. That is, the ***Request*** matrix is modified as follows:

	Request
	A B C
T_0	0 0 0
T_1	2 0 2
T_2	**0 0 1**
T_3	1 0 0
T_4	0 0 2

Answer whether the system below is in the deadlock state. If there does not exist a deadlock state, solve the problem using the algorithm presented above and write down the process in detail and find the sequence in which the system works without a deadlock. If there exists a deadlock, write down all the threads, consisting of the deadlock.