CS 330 - Assignment 7

Question 2:

a. Assume the number of nodes is n. Once all clients have sent their request, the number of messages sent is n. Before the one which has the lowest ticket numbers enters the critical section, it receives n - 1 replies from others. The one which has the second lowest ticket numbers will receive n - 2 replies from others (the one sent to the node which has lowest ticket no is deferred). Similarity, the third lowest ticket no node receive n - 3 replies and so on. So the total number of messages which has been sent is n + n - 1 + n - 2 + … + 1 = n * (n + 1) / 2.

b. According to Ricart-Agrawala algorithm, each node must have a unique ID. However, if there is a case in which several nodes have the same ticket numbers and ID, assume they want to enter the critical section at the same time, each of them will defer request messages from others which leads to none of them can enter the critical section ———> deadlock.

c. There is always a node which has the highest ticket numbers and that node will send replies to any node which sent request to it. Therefore its myDeferred list is empty.

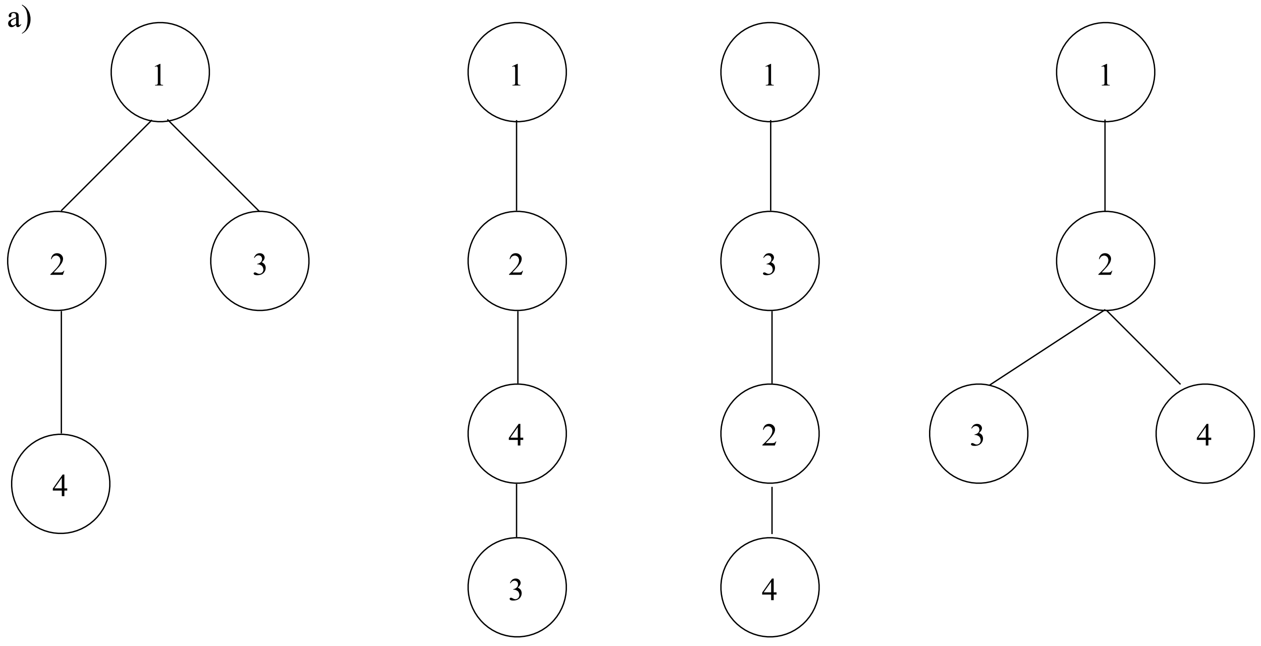
d. A node which has the lowest ticket number will defer all request from others except itself, which make the maximum number of entries in a single myDeferred list is n - 1.

e. When all nodes want to enter the cs, the lowest ticket no node's deferred list has n - 1 elements since it defers all request messages from other. The second lowest ticket no node's deferred list has n - 2 elements from others except the lowest ticket no node and so on. Therefore, the maximum number of entries is n-1 + n-2 + … + 1 = n * n / 2

f. After p13 finishes executing, if there is a node which has the lowest ticket number wants to enter the cs, it sends request messages to others, instead of getting replies, since myRequestCS is still set to true, its request will be deferred, which makes the algorithm incorrect.

Question 3:

a)



b)

Action	Node 1	Node 2	Node 3	Node 4
m 1->2	0	([0,0], 0)	([0,0,0], 0)	([0], 0)
m 1->3	1	([1, 0], 0)		
m 2->4	2		([1,0,0], 0)	
m 2->3		([1, 0], 1)		([1], 0)
m 2->4		([1, 0], 2)	([1,1,0], 0)	
s 4->2		([1, 0], 3)		([2], 0)
s 3->2		([1, 0], 2)		([1], 0)
s 4->2		([1, 0], 1)	([1,0,0], 0)	
s 3->1		([1, 0], 0)		([0], 0)
s 2->1	1		([0,0,0], 0)	
	0	([0, 0], 0)		
Termination was detected				

c) During termination of the algorithm, a node sends signals to all the nodes which sent message to it. Since the last signal is reserved to its parent, its parent's totalOutDeficit is always > 0 as long as its children's totalIndeficit > 0. Therefore, a parent node only terminate when all of its children terminate. Since the environment is root (it has no parent), the environment will terminate when all of general nodes in the network terminate. Therefore, the algorithm is live and safe.