Student's name:
Class:

Class Exercises
Module: Distributed Systems
Chapter 6: Synchronization (2/2)

Question 1: What is a mutual exclusion algorithm in a distributed system?

Question 2: What is the drawback of the centralized algorithm for the mutual exclusion?

Question 3: What is the drawback of the distributed algorithm for the mutual exclusion?

Question 4: Propose a solution for the problem of lost token in Token Ring mutual exclusion algorithm. (hint: you can improve/modify the original algorithm).

Question 5: A system of 8 nodes (P_0 to P_7) applies the Bully election algorithm. There are two broken nodes: P_4 and P_7. The node P_3 starts the election. How many messages does the system need to vote the coordinator?

Question 6: The nodes of a system are assigned an ID from *1* to *N*. Each node maintains a status table that consists of states of others nodes with two states: *Running* and *Broken*. An election algorithm is described as follows:
When a node P_i detects that the current coordinator is broken, it will look in the status table and send *ELECTION* message to the node whose ID just below the broken coordinator's ID. If the node that receives *ELECTION* still runs normally, it sends *OK* to P_i and then broadcasts the *COORDINATOR* message to all other nodes to inform that it is the new coordinator. If the node that receives *ELECTION* is broken, P_i continues to repeat the previous step with the nodes that have greater ID than himself (the information retrieved in the status table) until P_i receives *OK*, or the ID drops to its own ID. If the ID drops to its own ID, P_i knows that it becomes the new coordinator and broadcasts *COORDINATOR* message to the whole system.

a) How many messages do we need to vote the coordinator?
b) Propose an additional part for the above algorithm to handle the problem when a broken node comes back to running state. How many messages do we need for this recovery procedure? (hint: consider 2 cases where this node ID is greater and smaller than the current coordinator ID).

(* note: the answer related to the number of messages consists only two variables *N* and *i*. Do not put other variables into the result.)

Question 7: A mutual exclusion algorithm is described as follows:
Consider a system having *n* processes: P_1, P_2, ... P_n. There is a shared resource called SR (Shared Resource). Each process P_i maintains a queue *$queue_i$* to store requests that have not yet been executed.
When the process P_i wants to access the SR, it will broadcast a *REQUEST* message (ts_i, i) to all other processes, and store that message in its *$queue_i$* in which ts_i is timestamp of request.
When a process P_j receives a *REQUEST(ts_i, i)* from the P_i process, it takes that request into its queue ($queue_j$) and sends back to P_i a *REPLY* message.
The P_i process will allow itself to use SR when it checks that its request is located in the queue *$queue_i$* and other requests have a greater timestamp than its own.
After using SR, the process P_i deletes its request from the queue and broadcasts the *RELEASE* message for all other processes. When the process P_j receives a *RELEASE* message from P_i, it will delete the request of P_i in its queue.

a) How many messages does the system need to successfully let a process use SR?
b) There is an improvement the above algorithm described as follows: after P_j sends a *REQUEST* to other processes, it receives a *REQUEST* message from P_i, if it finds the its *REQUEST* timestamp is greater than the *REQUEST* timestamp of P_i, it won't send a *REPLY* message to P_i anymore. Does this improvement work? In applying this improvement, how many messages does the system need to successfully let a process use SR? Explain it.