NAME:_____SOLUTIONS_____

CS 425/ECE 428 Distributed Systems **Fall 2015**

Mid-Term Exam 1 (---- CORRECTED ----) **Total points: 100**

1. (8 points) This question relates to the "baseball" paper by Doug Terry.

   Suppose that process P1 performs the following write operations in the specified order:

   x := 1
   y := 2
   z : = 3

   Assume that, initially, all variables have value 0.
   Suppose that the above operations have been completed (i.e., P1 has received acknowledgements that these operations are completed).

   At a later time, process P1 performs the following operation:

   a := x + y

   Assume that no other operations performed.

   Under each of the consistency models below, what are the different values possible for variable *a*, after the above update of variable *a* has been completed?

   (a) Eventual consistency (i.e., without read-my-write guarantee)

   0, 1, 2, 3

   (b) Read-my-write

   3
   Note that P1 performs all the operations above.

2. (6 points) State true or false:

   (a) In a point-to-point synchronous network of 3 nodes, it is impossible to solve Byzantine Generals problem if one of the nodes may suffer Byzantine failure. TRUE

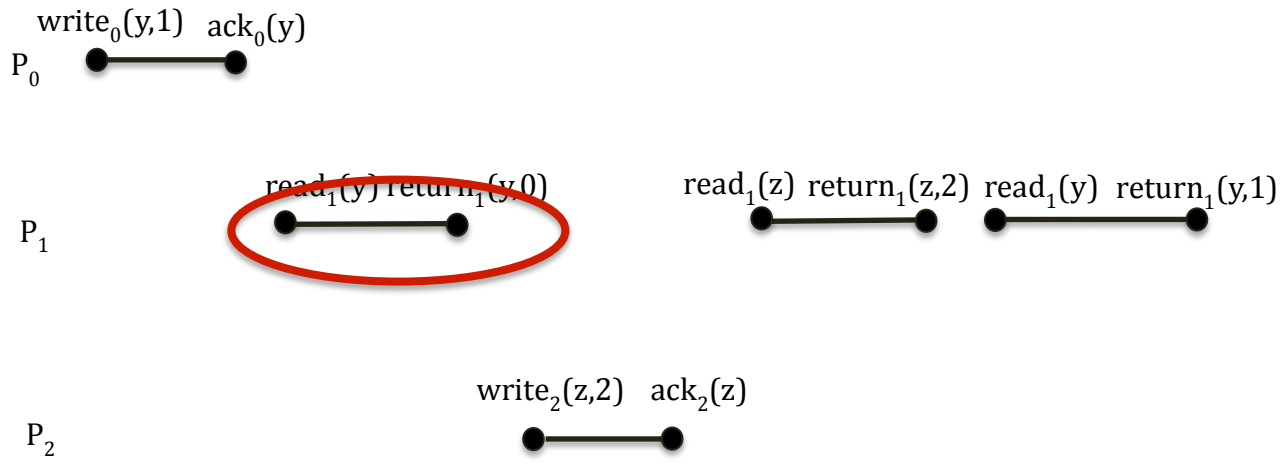
   (b) Suppose that V(e) and L(e) denote the vector and Lamport timestamps of event e, respectively. For any two events e and f, **if** V(e)<V(f) **then** L(e)<L(f). TRUE

   (c) Consensus is impossible in an asynchronous system wherein nodes may crash. TRUE

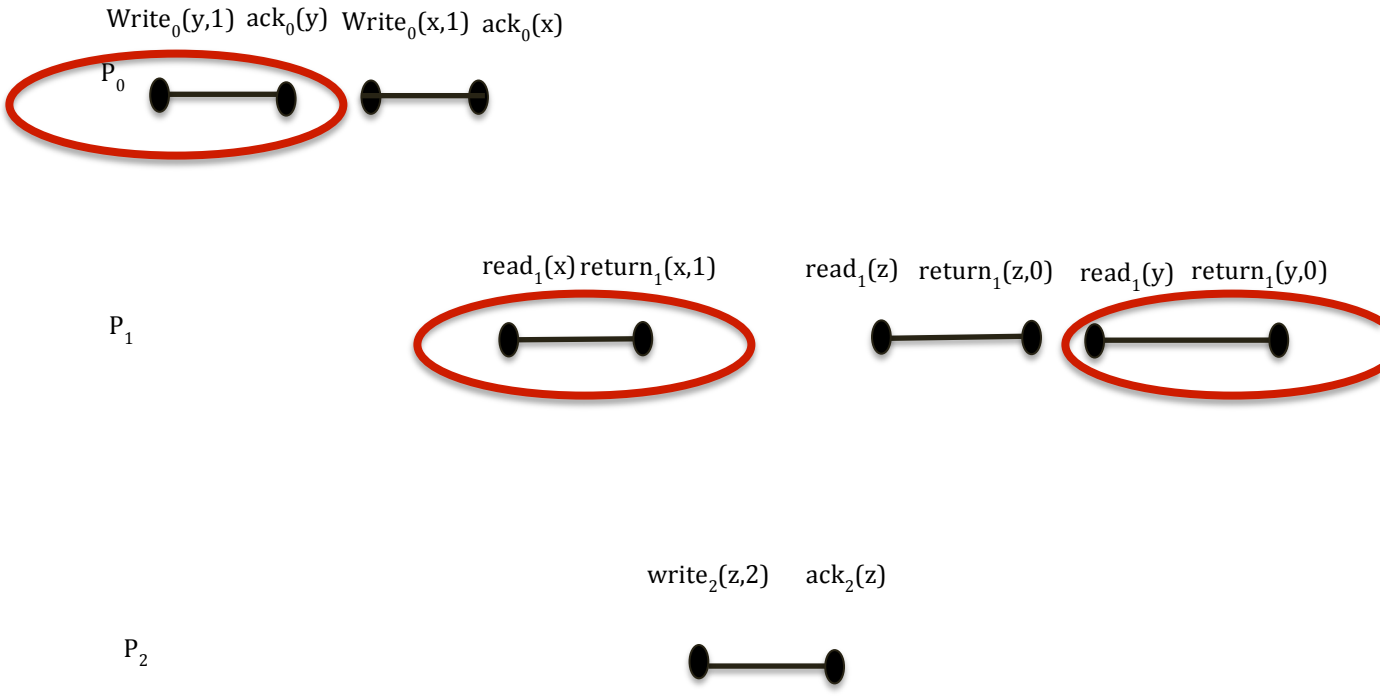
3. (15 points) In each part of this question, **if you answer NO, then delete a minimum number of operations to ensure that the modified execution will satisfy the specified property – circle the operations that you want to delete.**

Assume that all variables are initialized to **0**.

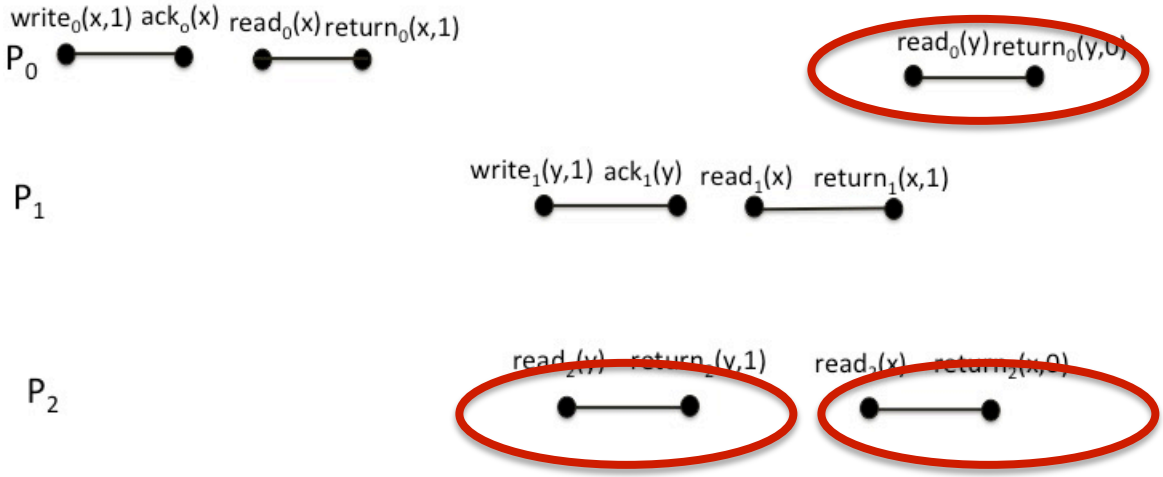
(a) Is the execution below linearizable? **NO**



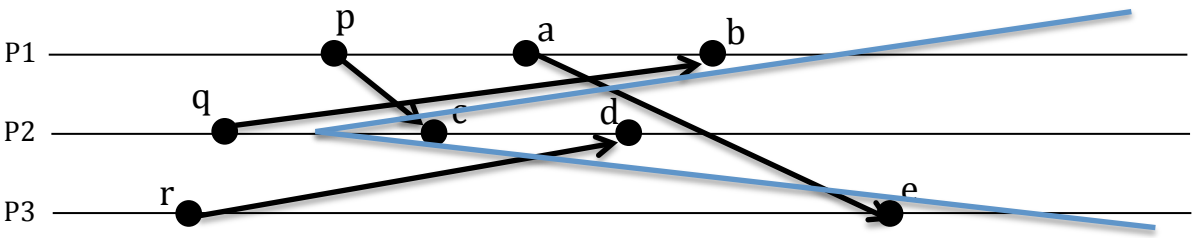
(b) Is the execution below sequentially consistent? **NO**
Any one of the circled pair of events can be removed to achieve sequential consistency.



(c) Is the execution below sequentially consistent?
No
<u>Any one</u> of the circled pair of events can be removed to achieve sequential consistency.



4. (10 points)
(a) In the execution below, draw the consistent cut that contains the largest possible number of events, ***excluding*** **event c**.

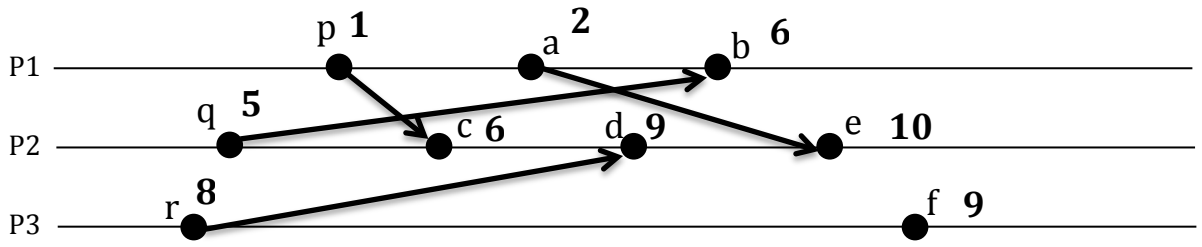


A cut containing {a, b, e, p, q, r}

(b) In the execution above, is the cut consisting of events {a.b,e,p,q,r} a consistent cut? *Explain your answer*.
Yes.
No messages that are sent from the right side of the cut are received on the left side. More formally, for each pair of events x and y, if x → y and y is in the cut, then x is also in the cut.

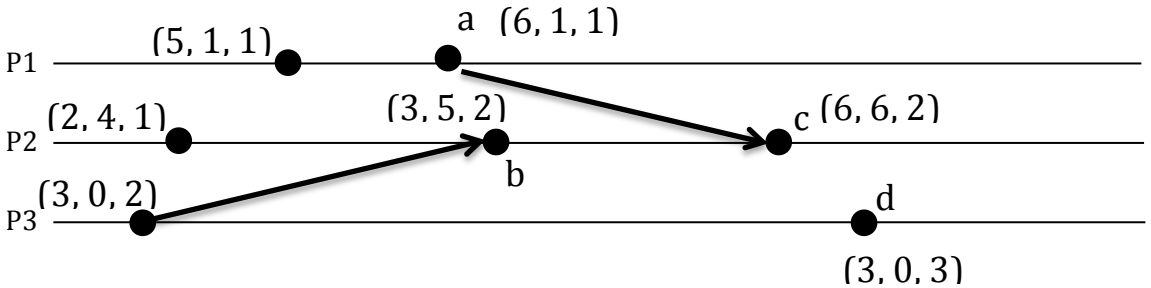
5. (12 points)

(a) In the execution below, assume that logical timestamps of **events p, q and r are 1, 5 and 8**, respectively. Determine the logical timestamps of the remaining events in the figure. Write the logical timestamp of each event next to that event in the figure.

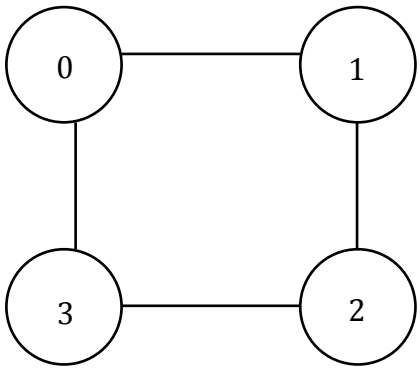


(b) In the execution shown ABOVE, determine all the events that are concurrent with event d.
{a,b,f}

(c) The figure below shows the vector timestamp of 3 events in the execution. Determine the vector timestamps for the remaining events shown below. Write the vector timestamps of each event next to that event in the figure.



6. (12 points) Consider a synchronous system shown below. The system consists of four nodes (named 0, 1, 2 and 3) and 4 bidirectional links.



(a) Assume the following: (i) Nodes 0 and 1 never fail, and (ii) nodes 2 and 3 may both crash, either simultaneously or individually.
Is it possible to solve consensus in this system? If you answer no, explain why.
If you answer yes, provide a sketch of an algorithm to solve consensus on this system.

YES.
0 sends its input directly to nodes 1 and 3 using the links to them. Node 1 forwards node 0's input to node 2. All nodes set their output to node 0's input thus received.

The algorithms works correctly because nodes 0 and 1 never fail.

(b) Assume the following: (i) Nodes 0 and 2 never fail, and (ii) nodes 1 and 3 may both crash, either simultaneously or individually.
Is it possible to solve consensus in this system? If you answer no, explain why.
If you answer yes, provide a sketch of an algorithm to solve consensus on this system.

No. The solution is similar to that of problem 3 on homework 3.

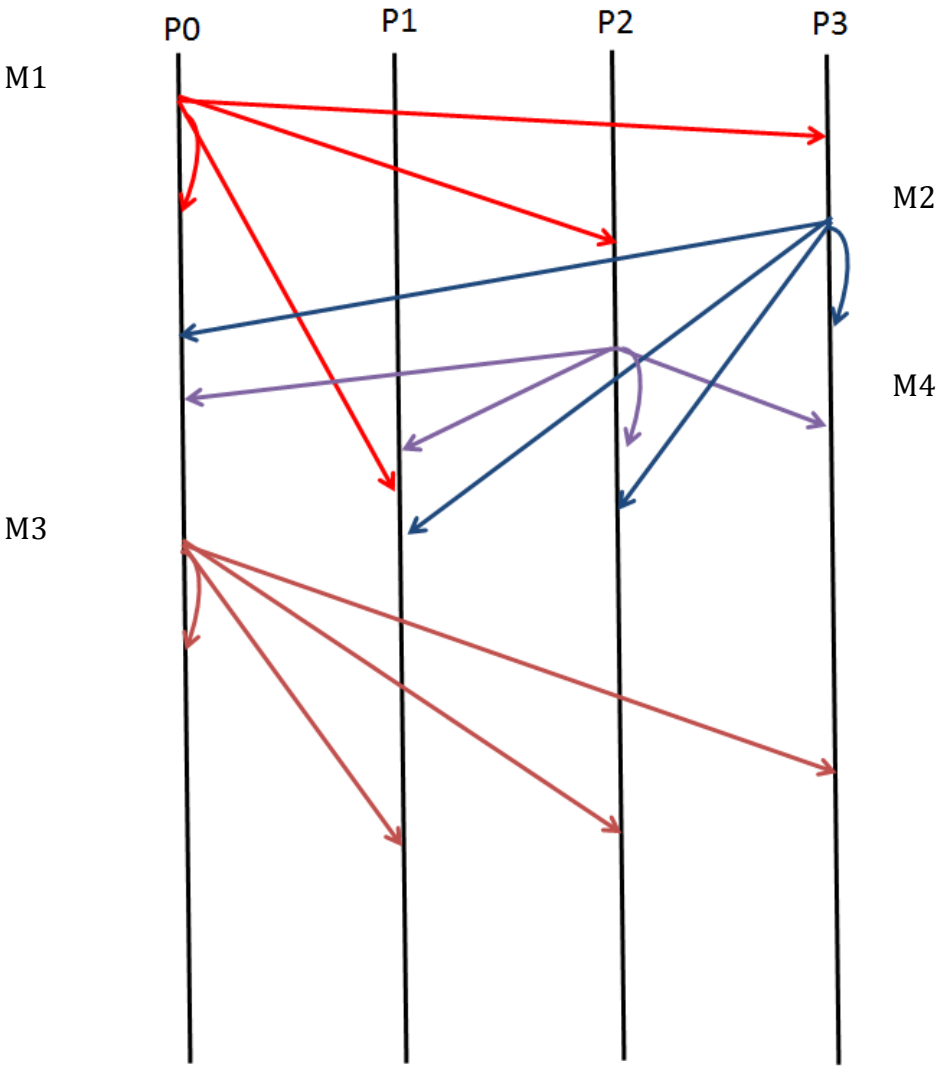
7. (a) (6 points) The execution below shows the time at which four multicast messages are delivered to processes P0, P1, P2 and P3. Does the message deliveries satisfy the following properties? In each case, answer Yes or No -- if you answer No, explain why.

(i) FIFO ordering
Yes

(ii) Causal ordering

No, send(M1)→send(M4) yet P1 receives M4 before it receives M1.

(iii) Total ordering
No, order at P1: M4, M1, M2, M3, while order at P0: M1, M2, M4, M3



(b) (7 points) The algorithm below is a modified version of the algorithm in the *last slide* in the handout provided to you for this exam. The modification is to delete a part of the code in the "wait until" statement, as shown below. The original (unmodified) algorithm achieves causally ordered multicast.

Does the modified algorithm achieve any of the following properties for the multicast? For each property, answer Yes or No.

(i) FIFO ordering
Yes

(ii) Causal ordering
No

(iii) Total ordering
No

Algorithm for group member p_i $(i = 1, 2\ldots, N)$

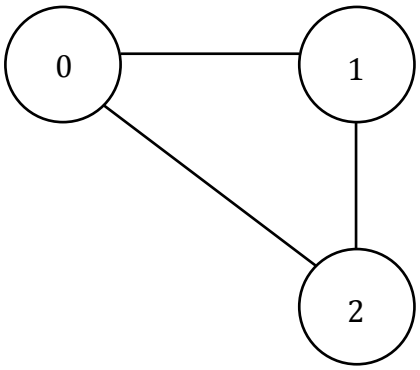
On initialization
$V_i^g[j] := 0\ (j = 1, 2\ldots, N)$;

The number of group-g messages from process j that have been seen at process i so far

To CO-multicast message m to group g
$V_i^g[i] := V_i^g[i] + 1$;
B-multicast$(g, <V_i^g, m>)$;

On B-deliver$(<V_j^g, m>)$ *from* p_j, *with* $g = group(m)$
place $<V_j^g, m>$ in hold-back queue;
wait until $V_j^g[j] = V_i^g[j] + 1$ ~~and $V_j^g[k] \leq V_i^g[k]\ (k \neq j)$~~;
CO-deliver m; // after removing it from the hold-back queue
$V_i^g[j] := V_i^g[j] + 1$;

8. (12 points) Consider the systems shown in the figure below. Assume the following: (i) the message delay on link between nodes 0 and 1 is unbounded (maybe arbitrarily large), and (ii) on each of the remaining links, the message delay is always less than 50 ms. You may assume that processing delays are 0.



(a) Is it possible for the three nodes to synchronize their clocks to within 100 ms of each other? (i.e., skew <= 100 ms). Answer YES or NO.
If you answer NO, explain why.
If you answer YES, sketch an algorithm for achieving the above objective.

YES. Nodes 0 and 1 can each synchronize with node 2 using Cristian's or NTP algorithm. Due to the maximum 50 ms delay (one-way delay), each of nodes 0 and 1 will achieve skew of at most 50 ms with respect to node 2 (as per the analysis in the book). Therefore, nodes 0 and 1 will achieve skew of at most 100 ms relative to each other.

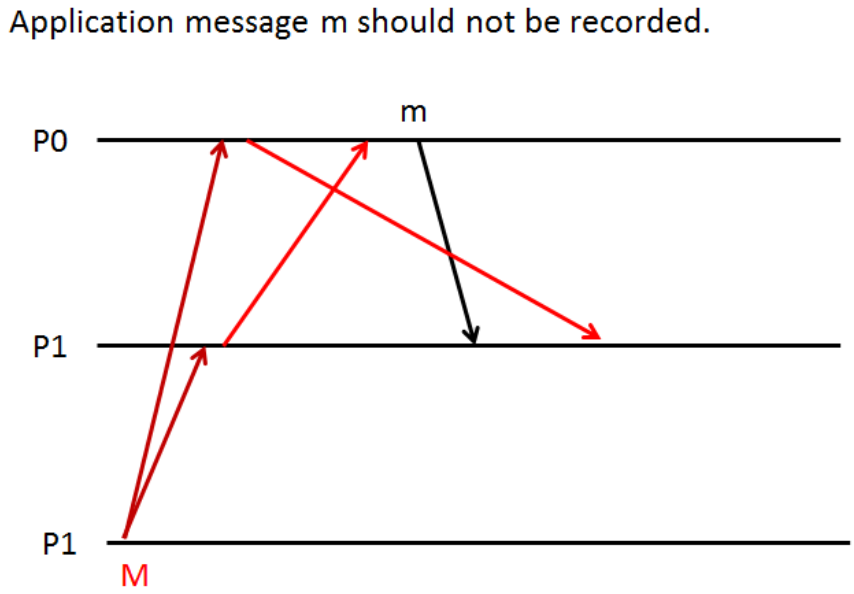
The above algorithm does not use the link between nodes 0 and 1.

(b) If we also know that message delays are always at least 10 ms, is it possible for the three nodes to synchronize their clocks to within 90 ms of each other? (i.e., skew <= 90 ms). Answer YES or NO.

YES.

9. (6 points) Consider the Chandy-Lamport algorithm on page 2 of the handout provided to you for this exam. Will this algorithm record a correct snapshot if the communication channels are not FIFO? Justify your answer.

No. A correct snapshot needs to record both correct local state and correct channel state. Consider the following example: In the figure, red messages are markers. Notice that message m of P0 reaches P1 before the marker from P0, even though it was sent after the marker. In this case, process P1 will **incorrectly** record m as the state of the channel from P0 to P1.



10. (6 points) Does there exist any algorithm to achieve approximate consensus in an asynchronous system with up to f crash failures, if the total number of nodes is f+1. Justify your answer.

No. There does not exist any algorithm that can achieve approximate consensus in asynchronous system with n = f+1 .

Consider f+1 processes named p0, p1, …, pf.
Consider the case when f process pf has input 1, and the remaining f processes have input 0.
Suppose that the messages between pf and the remaining f processes, take a very long time to be delivered.

For any algorithm, process pf has two cannot distinguish between the following two scenarios:
1 The other f processes have crashed before sending any messages to pf.
2, The messages from the other f processes are just delayed.
In the first scenario, process pf must decide on ouptut 1 (i.e., output equaling its own input).
Since the two scenarios can look identical to pf, it must decide on 1 in scenario 2 as well.

However, by a similar argument, the other f processes must decide on 0 (since all of them have input 0).

This implies that consensus on an identical value is not guaranteed even when all processes are fault-free.