This part for lecturer's grading use only

NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
AY2013/2014, SEMESTER 2

FINAL EXAMINATION FOR CS4231: Parallel and Distributed Algorithms

Lecturer: YU Haifeng

April 2014 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

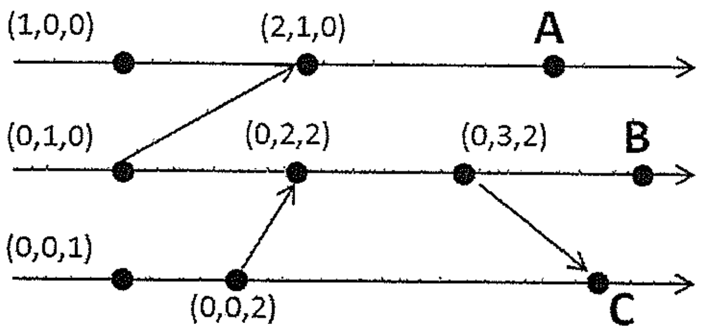
1. This examination paper contains **TEN** problems and **TEN** printed pages, including this page.
2. Students are required to answer **ALL** questions.
3. All questions should be answered within the space provided in this booklet.
4. This is an **Open Book** examination. Electronic calculators are **not** allowed.
5. Read each question carefully before you answer. If you misunderstand the question and answer a different question, you get **zero credit** for that question.
6. It is your job to show the correctness of each statement you make. Otherwise points will be deducted **even if the lecturer cannot prove you are wrong**. (This is a general rule for all theory subjects – otherwise you can claim P=NP and nobody can prove you are wrong.)
7. Write your answers clearly. **Unreadable handwriting will not be graded and will earn you zero points.**
8. Write your matriculation number below. Do not write your name.

MATRICULATION NO:

DO NOT WRITE BELOW THIS LINE

Problem 1 and 2	
Problem 3 and 4	
Problem 5	
Problem 6	
Problem 7	
Problem 8	
Problem 9	
Problem 10	
Total	

Problem 1. (6 marks)



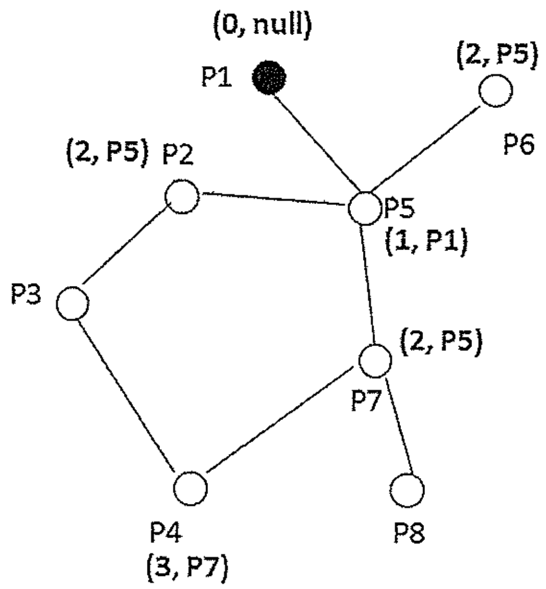
Imagine that we run the standard vector clock protocol in the above execution of a certain distributed system. The figure has already indicated the vector clock values of all the events, except for events A, B, and C.

What should be the vector clock value for event A? ______________

What should be the vector clock value for event B? ______________

What should be the vector clock value for event C? ______________

Problem 2. (4 marks)



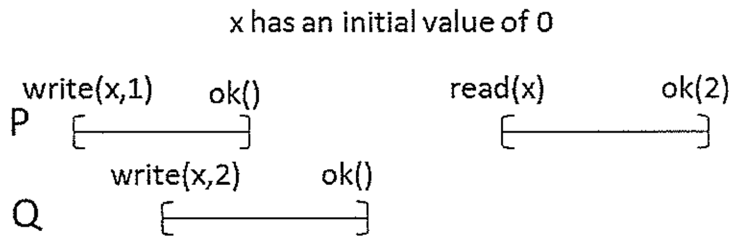
Consider the self-stabilizing spanning tree algorithm for maintaining a spanning tree in a distributed system. Imagine that we run this algorithm in the above topology, **until everything stabilizes**. The figure has already indicated the (dist, parent) tuple for all the nodes after everything stabilizes, except for nodes P3 and P8.

On node P3, what should be the value for the (dist, parent) tuple after everything stabilizes? ______________

On node P8, what should be the value for the (dist, parent) tuple after everything stabilizes? ______________

This part for lecturer's
grading use only

Problem 3. (4 marks)



Consider the above history of an execution of a parallel system.

Is the above history sequentially consistent? (just indicate "yes"/"no", no need to explain why)_____________

Is the above history linearizable? (just indicate "yes"/"no", no need to explain why)_____________

Problem 4. (4 marks)
Consider the Rotating Privilege algorithm for maintain a privilege on a ring. In that algorithm, one node is Red, and all other nodes are Green. Clearly explain what will go wrong if we change the number of Red and Green nodes in the following two ways, respectively:

What will go wrong if all nodes are Green?

What will go wrong if two nodes are Red, and all other nodes are Green?

Problem 5. (6 marks)
Consider an anonymous system with n nodes (i.e., the nodes do not have unique ids, and they all start with the same state and run the same algorithm), where the timing model is synchronous. We know that if these n nodes form a ring topology, then it is impossible to design a deterministic leader election algorithm for them. Now consider a different setting where these n nodes are wireless nodes, and whenever one node sends a message, all nodes will receive that message. In other words, they form a clique topology (i.e., complete graph). Assume that there is no collision (i.e., it is possible for many nodes to send message simultaneously). Do you think it is possible to design a **deterministic** leader election algorithm in this new setting? If you believe that it is possible, then i) indicate "Yes", ii) provide a high level intuition of how your leader election algorithm works, iii) provide concise pseudo-code for your algorithm, and iv) prove that the your algorithm is correct. If you believe that it is not possible, then i) indicate "No", ii) explain clearly why this cannot be done. Only indicating "Yes"/"No" will not earn you any mark.

First, do you believe that it is possible? (Yes/No) ______________

Next, please provide a detailed answer according to the above instructions:

This part for lecturer's grading use only	CS4231

Problem 6. (8 marks)
Use Java monitor to implement the notion of *K-robust critical sections*, where there can be up to *K* but no more than *K* processes being the critical section at the same time. (Here *K* is known beforehand.) Note that 1-robust critical section is the same as the standard critical section that we learned in class. You should write out the pseudo-code for the two methods EnterKCS() and ExitKCS() clearly.

First, write any shared data objects you will use and indicate their initial values below:

Next, write out the pseudo-code for the void EnterKCS(int process_id) method below, using Java monitor:

Finally, write out the pseudo-code for the void ExitKCS(int process_id) method below, using Java monitor:

This part for lecturer's grading use only

Problem 7. (8 marks)
Consider any set T of mutually-concurrent events in any execution of any distributed system. (Namely, no event in T happened before any other event in T.) Let T1 be any subset of T, and let T2 = T−T1. (Namely, T2 is obtained by subtracting T1 from T.) Alice claims that **for any T, T1, and T2 in any execution**, there always exists a consistent global snapshot S such that S contains all events in T1, and does not contain any event in T2. Do you agree with Alice?

If you agree with Alice, then i) indicate "Yes", ii) explain clearly how such S can be constructed, iii) prove that S contains all events in T1 and does not contain any event in T2, iv) prove that S is indeed a consistent global snapshot. If you do not agree with Alice, then i) indicate "No", ii) construct a counter-example which should include an example execution of a distributed system, the sets T, T1, and T2, iii) explain clearly why it is impossible to find S in your counter-example. Only indicating "Yes"/"No" will not earn you any mark.

Do you agree with Alice? (yes/no) ____________________

Next, give a detailed answer according to instructions above:

This part for lecturer's grading use only	CS4231

Problem 8. (8 marks)
Consider a distributed system where every message has a propagation delay of **exactly** one second. Assume that each received message is delivered immediately to the receiving process, without any extra delay. Also assume that a sending event itself does not incur any delay. Alice claims that the messages in this distributed system always satisfy causal order. Do you agree with Alice?

If you agree with Alice, then i) indicate "Yes", ii) provide a proof showing that the messages in this system always satisfies causal order (your should describe the high-level idea of your proof first). If you do not agree with Alice, then i) indicate "No", ii) construct a counter-example which should include an example execution of a distributed system, iii) clearly indicate which two messages in your counter-example violates causal order. Only indicating "Yes"/"No" will not earn you any mark.

Do you agree with Alice? (yes/no) ____________________________

Next, give a detailed answer according to instructions above:

This part for lecturer's grading use only

Problem 9. (6 marks)
Consider the byzantine generals problem (namely, the distributed consensus problem where nodes may experience byzantine failures, every node can directly communicate every other node, all communication channels are reliable, and the timing model is synchronous). Let *n* be the total number of nodes, and *f* be the maximum number of nodes that can experience byzantine failures. In class, we learned the following theorem, but we did not discuss the proof for this theorem:

Theorem 1. The byzantine generals problem is not solvable when $n \leq 3f$.

You are now asked to prove the following Theorem 2, which is a special case of Theorem 1:

Theorem 2. The byzantine generals problem is not solvable when $n=6$ and $f=2$.

Your proof CANNOT invoke Theorem 1. However, you can assume that the following theorem (Theorem 3) holds, and invoke it in your proof:

Theorem 3. The byzantine generals problem is not solvable when $n=3$ and $f=1$.

To reiterate, you are asked to prove Theorem 2. Your proof CANNOT invoke Theorem 1, but can invoke Theorem 3. (Hint: Try constructing a proof by contradiction.)

Write your proof below:

This part for lecturer's grading use only

Problem 10. (6 marks)
Consider a synchronous distributed system with n nodes, where n is odd and every node knows the value of n. The topology among these n nodes can be any connected undirected graph. Each node has a **distinct** integer value between 0 and n^2, as the node's input. Assume that there is a distinguished node Z in the system. We learned in class that by building a spanning tree in the network, Z can efficiently figure out the max, min, sum, and average of these n values. You are now asked to design a distributed algorithm for Z to figure out the **median** of all these n inputs. (Hint: Try using the sum algorithm as a building block of your algorithm.) In your algorithm, each node is allowed to send at most $O(b)$ bits to each of its neighbors, with b being some polynomial of a where a=log(n). **Remember that it is your job to explain clearly how your algorithm works, why it is correct, and why it satisfies all the requirements. Vague or unclear descriptions will result in penalties.** Give you answer in **the following 4 Steps.**

Step 1. Explain clearly the high-level idea of your algorithm:

Step 2. Write out the **high-level** pseudo-code of your algorithm:
(Please start your pseudo-code on the next page.)

3. Explain clearly why your algorithm computes median correctly:

4. Explain clearly why in your algorithm, each node sends at most $O(b)$ bits to each of its neighbors:

END-OF-PAPER