This part for lecturer's grading use only	CS4231

NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

AY2015/2016, SEMESTER 2

FINAL ASSESSMENT FOR

CS4231: Parallel and Distributed Algorithms

25 APR 2016 **Time Allowed: 2 Hours**

INSTRUCTIONS TO CANDIDATES

1. This assessment paper contains **ELEVEN** problems and **EIGHT** 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** assessment. 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 student number below. Do not write your name.

STUDENT NO:

DO NOT WRITE BELOW THIS LINE

Problem 1, 2, 3	
Problem 4, 5, 6	
Problem 7, 8, 9	
Problem 10	
Problem 11	
Total	

Problem 1. (4 marks)
A, B, and C are three events in a certain execution of a certain distributed system. We already know that A happened before B, and B happened before C.

If the logical clock values of A and C are 5 and 8, respectively, what are the possible logical clock values for B? (You only need to write out the possible values, no explanation needed.)

_____

Problem 2. (4 marks)
Give an example execution of an example distributed system with two events A and B, such that A does not happen before B and B does not happen before A. If you feel that there does not exist such an execution, then prove the non-existence of such executions.

Problem 3. (4 marks)

Initial value of flag = 0;

```
Code for P1:                Code for P2:
Part 1;                     Part 3;
while (flag == 0) {         flag = 1;
   do nothing;              Part 4;
}
Part 2;
```

In the above code segment, Alice tries to coordinate two processes P1 and P2, so that P1 will not move on to "Part 2" until P2 has first finished executing "Part 3". Do you feel that Alice achieves her goal? Briefly explain why. If you feel that Alice does not achieve her goal, further explain how to modify the code to achieve her goal.

Problem 4. (4 marks)
Give an example execution of an example distributed system with 3 processes, where the message delivery (for broadcast messages) satisfies causal order but not total order. If you feel that there does not exist such an execution, then prove the non-existence of such executions.

Problem 5. (4 marks)
Consider any history that can result from the execution of a single-process (and also single-threaded) program on a typical shared-memory multi-processor computer.

Does such a history always satisfy linearizability? (Yes/No) _____

Does such a history always satisfy sequential consistency? (Yes/No) _____

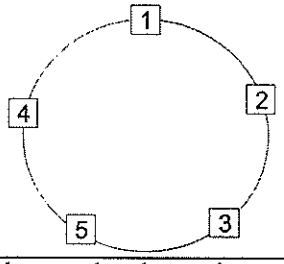
Problem 6. (8 marks)
We know that in a certain execution of a certain distributed system, there are total n events. We use X to denote the number of different consistent global snapshots that this execution has. (Note that a consistent global snapshot is allowed to be an empty set, so the empty set also counts toward X.)

What is the maximum possible X? _____
Please describe an example execution that corresponds to such X:

What is the minimum possible X? _____
Please describe an example execution that corresponds to such X:

Problem 7. (4 marks)
Not every problem can be deterministically solved in all distributed systems. Give an example problem that can be deterministically solved in synchronous distributed systems but not in asynchronous distributed systems. If you feel that there does not exist such a problem, then prove the non-existence of such problems.

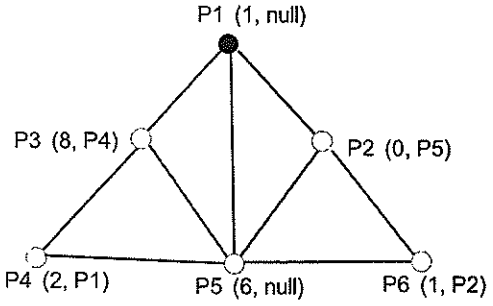
Problem 8. (4 marks)



Imagine that we run Chang-Roberts algorithm on the above ring. How many messages will the algorithm incur in total? (Just write the answer, no explanation needed.)

_____

Problem 9. (8 marks)
Consider the self-stabilizing spanning tree algorithm. Assume that the system starts from the following state, where P1 is the root:



In the figure, the tuple beside each node indicates the current values of the `dist` variable and the `parent` variable on that node.

Starting from the above state in the figure, what is the maximum number of times that P2 may change the value of its `dist` variable? _____

Give one example sequence of actions taken by the nodes over which P2 will change the value of its `dist` variable for so many times:
(For example, a sequence of P1 P2 P3 P2 means that P1 takes an action (i.e., execute its code once), then P2 takes an action, then P3 takes an action, then P2 takes an action.)

_____

Problem 10. (8 marks)

Consider the distributed consensus problem where the timing model is **synchronous**, the communication channels are reliable, and the nodes may experience byzantine failures. There are total $n = 4f+1$ nodes (with ids from 0 through $n-1$), and the maximum number of nodes that may experience byzantine failures is f. Every node can directly communicate with every other node. Whenever a node receives a message, it can always correctly determine the sender of the message. Every node has a binary initial value. We learned in class a protocol for solving the consensus problem under the above setting, where the protocol has $f+1$ phases, and each phase has 2 rounds. So the total time complexity is $O(f)$ rounds.

Now we change the problem in the following way:

-- $n = 8f$, and f is a power of 2.

-- There is a special Oracle. At the beginning of the execution, the Oracle generates an infinitely-long random bit string S, and gives S to all n nodes. (The Oracle never experiences any failure.) Here S is independent of i) the inputs to the nodes, ii) the behavior of the scheduler, and iii) which nodes experience byzantine failures.

-- With probability at least 15/16, your algorithm should satisfy the standard definitions of *agreement* and *validity*. Here the probability is taken over S. Note that when reasoning about this probability, you need to consider the worst-case input and worst-case scheduler.

You are asked to design a deterministic consensus protocol (which takes S as one of its inputs) to solve the above consensus problem (with the 3 new conditions). You should minimize the asymptotic time complexity (in terms of the number of rounds) of your protocol. The marks you obtain will depend on the how small your asymptotic time complexity is. Please provide your answers in the following steps:

Step 1. The asymptotic time complexity of your protocol is O(_____) rounds.

Step 2. Provide high-level intuition of your protocol:

Step 3. Give pseudo-code of your protocol:

(Problem continues onto the next page....)

Step 4. Rigorously prove that your protocol satisfies all the properties needed (including proving the probability of satisfying agreement/validity).

Problem 11. (8 marks)
Consider the distributed consensus problem where the timing model is **asynchronous**, the communication channels are reliable, and the nodes may experience crash failures. There are total N nodes, and the maximum number of nodes that may experience crash failures is one. Every node can directly communicate with every other node. Every node has a binary initial value. Each node has a local random number generator which it can use to generate random numbers. Random numbers generated on different nodes are independent. You are asked to design a randomized algorithm to solve the distributed consensus problem under such a setting, while satisfying all the following 3 properties:

-- Your algorithm should always *terminate*.

-- With probability at least 1-t where $t < 1$, your algorithm should be correct in the sense that it satisfies the standard definitions of *agreement* and *validity*. Here the probability is taken over all the random numbers generated by all the nodes. Note that when reasoning about this probability, you need to consider the worst-case input and worst-case scheduler. You should try to achieve, asymptotically, as small a t value as possible (i.e., t should be written as a function of N, and t's asymptotic form is the asymptotic form of the function when N grows toward infinity). The marks you obtain will depend on how small your t is asymptotically.

-- Your algorithm, if it were used instead in a standard synchronous system without any modification, should always terminate within O(1) rounds. (Note that even though your algorithm is designed for the above asynchronous setting, we can nevertheless use it in synchronous systems.)

The following is a **fact**, which you may choose to use or not use: Let X be the random variable denoting the number of heads obtained when flipping K independent fair coins (note that the probability of having a head when flipping one fair coin is 0.5), then Pr[X = i] = O(1/sqrt(K)) for all i from 0 to K.
Please provide your answers in the following steps:

Step 1. What is the asymptotic value of t that your protocol achieves? t = O(_____)

Step 2. Provide high-level intuition of your protocol:

Step 3. Provide non-ambiguous pseudo-code of your protocol:

(Problem continues onto the next page....)

Step 4. Rigorously prove that your protocol satisfies all the properties needed (including proving the probability of satisfying agreement/validity):

END-OF-PAPER