This part for lecturer's grading use only

NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

AY2017/2018, SEMESTER 2

FINAL ASSESSMENT FOR

CS4231: Parallel and Distributed Algorithms

02 May 2018 **Time Allowed: 2 Hours**

INSTRUCTIONS TO CANDIDATES

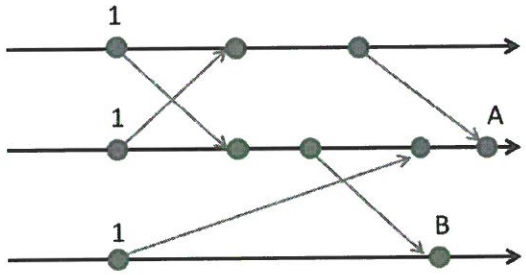
1. This assessment paper contains **TWELVE** problems and **TEN** printed pages, including this page.
2. Students are required to answer **ALL** questions.
3. All questions should be answered within the space provided in this booklet.
4. This is an **Open Book** assessment.
5. Read each question carefully 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	
Problem 3,4	
Problem 5,6	
Problem 7,8	
Problem 9	
Problem 10	
Problem 11	
Problem 12	

Problem 1 (4 marks)

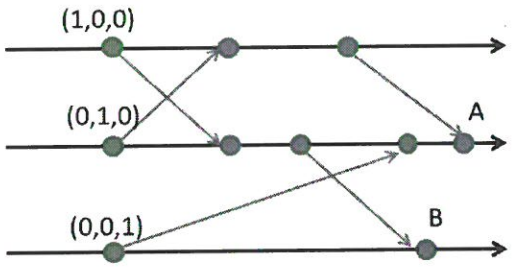


Consider the above execution of a distributed system, where the logical clock value of the first event on each process has already been indicated.

What is the logical clock value for event A? (Just write the value, no explanation needed.) _____

What is the logical clock value for event B? (Just write the value, no explanation needed.) _____

Problem 2 (4 marks)



Consider the above execution of a distributed system, where the vector clock value of the first event on each process has already been indicated.

What is the vector clock value for event A? (Just write the value, no explanation needed.) _____

What is the vector clock value for event B? (Just write the value, no explanation needed.) _____

Problem 3 (6 marks)
In the context of a parallel system, imagine that 3 processes execute the program "x = 2*x" in parallel. Here x is a shared variable whose initial value is 1. **After all 3 processes complete their executions:**

Is it possible for x to have a value of 4? (Just indicate Yes/No, no further details needed) _____

Is it possible for x to have a value of 2? (Just indicate Yes/No, no further details needed) _____

Is it possible for x to have a value of 1? (Just indicate Yes/No, no further details needed) _____

Problem 4 (6 marks)
Consider the following mutual exclusion algorithm based on the special TestAndSet machine-level instruction:

```
Shared integer variable Locked initialized to 0;

RequestCS(process_id) {
    while (Locked.TestAndSet(1) = = 1) {;}
    // Locked.TestAndSet(1) will set the value of Locked to 1 and
    // then return the old value of Locked, in one atomic step.
}

ReleaseCS(process_id) {
    Locked = 0;
}
```

Does this algorithm satisfy the No-Starvation property?

Answer "Yes"/"No" first: _____

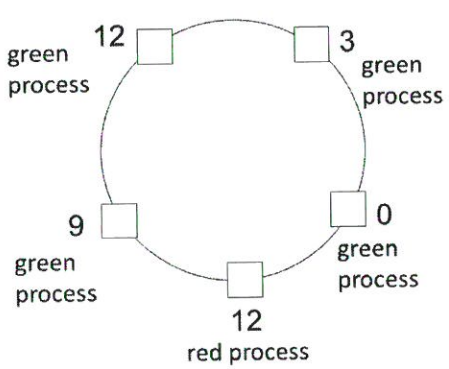
If you indicated "Yes", please give a rigorous proof. If you indicated "No", please construct a counter-example. Only answering Yes/No without further details will not earn you any marks.

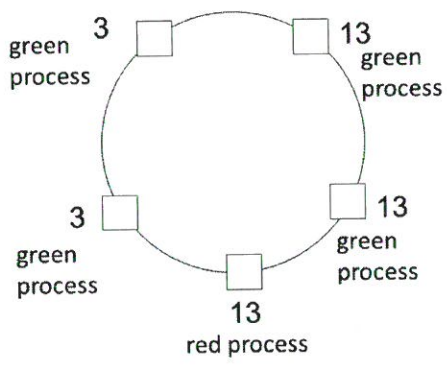
Problem 5 (6 marks)
Consider a distributed system where the nodes have perfectly accurate physical clocks. In such a system, if every node takes a local snapshot at exactly the same physical time, will the union of the local snapshots always be a consistent global snapshot?

Answer "Yes"/"No" first: _____

If you indicated "Yes", please give a rigorous proof. If you indicated "No", please give a counter-example. Only answering Yes/No without further details will not earn you any marks.

Problem 6 (4 marks)
Consider the rotating privilege algorithm learned in class, which is a self-stabilizing algorithm for solving the rotating privilege problem on a ring. Let n = 5 and k = 15 in this algorithm, and consider the following two states of the system.





Is it ever possible for the system to start from the state on the left, and then reach the state on the right? (Just indicate Yes/No, no further details needed) _____

Problem 7 (4 marks)
Consider the distributed algorithm that we learned in class for constructing a spanning tree in a network with some arbitrary connected and undirected topology. If the topology has N nodes, what is the best-case and worst-case message complexity of this algorithm? Here the best-case (worst-case) is taken over the best-case (worst-case) connected and undirected topology.

The best-case message complexity (in its asymptotic tight form with respect to N) is $\Theta($_____$)$

The worst-case message complexity (in its asymptotic tight form with respect to N) is $\Theta($_____$)$

Problem 8 (6 marks)
Alice has designed the following algorithm for ensuring causal order message delivery. The system maintains standard logical clocks. When a node sends a message, the node will attach the logical clock value of the send event on the message. We will call this value as the message's *priority value*. Time is divided into intervals of fixed duration, and we assume that all nodes have perfectly accurate physical clocks. For any node X, it will buffer all messages received during the current interval. At the end of the current interval, X will order all the messages in its buffer, according to increasing priority values. X will then deliver all the messages in its buffer, by this ordering, to the application. After that, X will move on to the next interval.

Do you feel Alice's algorithm is correct? (Yes/No) _____

If you indicated "Yes", please give a rigorous proof. If you indicated "No", please give a counter-example. Only answering Yes/No without further details will not earn you any marks.

Problem 9 (4 marks)
Consider the distributed consensus problem where the timing model is synchronous, the channels are reliable, and the nodes may experience crash failures. There are total n nodes with distinct ids, and the maximum number of nodes experiencing crash failures is 2. Alice has designed the following protocol for solving the problem. The protocol has 2 rounds. In the first round, each node X will send the tuple (X's id, X's input value) to every node (including itself). In the second round, each node X will send the tuple (X's id, X's input value), as well as all the tuples it received in the first round, to every node (including itself). At the end of the second round, each node X will examine the set of distinct tuples that it has seen so far, and let (a, b) be the tuple whose first field (i.e., a) is the smallest, among all tuples in that set. Node X will then decide on the value b.

Do you feel Alice's algorithm is correct? (Yes/No) _____

If you indicated "Yes", please give a rigorous proof. If you indicated "No", please give a counter-example. Only answering Yes/No without further details will not earn you any marks.

Problem 10 (6 marks)
Consider leader election on rings with $n \geq 100$ nodes, where n is known to you. We will assume that the timing model is synchronous. The ring is bidirectional, which means that a node can send messages to its clockwise neighbor as well as to its counter-clockwise neighbor. When sending a message, a node is able to choose whether to send it clockwise or counter-clockwise. (In order words, a node can tell which of its two incidental edges is its clockwise outgoing edge.) Among the n nodes, n-1 nodes have the same id, while the single remaining node has a different id. For example, when n=100, we may have 99 nodes with id 72 and 1 node with id 29. Please design a deterministic leader election algorithm to elect a leader, such that the leader will output "LEADER" while all other nodes should output "NON-LEADER". You should minimize the worst-case message complexity (i.e., total number of messages incurred) of your algorithm. We will be concerned with the **exact form** of the complexity, rather than the asymptotic form. (**Hint: Your algorithm should not start with something like "the node with the unique id will do XXX", since initially, a node does not know whether it is the node with the unique id.)**

What is the exact (not the asymptotic form) worst-case message complexity (i.e., total number of messages incurred) of your algorithm, as a function of n? _____

Give your algorithm below. You are allowed to invoke any algorithm learned in class if you want, but you should clearly indicate which algorithm you invoke, and you should invoke the algorithm as a black-box. Your answer should have 4 parts i) a high-level intuitive description of your central idea, ii) precise pseudo-code for your algorithm, iii) correctness proof of your algorithm, and iv) proof for the worst-case message complexity of your algorithm.

Problem 11 (4 marks)
Consider the distributed consensus problem where the timing model is asynchronous, the channels are reliable, and the nodes may experience crash failures. There are total n nodes, where n is larger than 10000, n is a multiple of 20, and n is given to you. The total number of nodes that crash is at most n/20. For this problem, we will require that a node stops sending messages once it has decided. Recall that the **Agreement** requirement in the distributed consensus problem requires that "**all nodes that decide (including those who decided and then crashed) should decide on the same value**". In this problem we will replace this **Agreement** requirement with a **Weakened-Agreement** requirement, defined as following: "**Let k be the total number of nodes that have decided (including those who decided and then crashed) by the end of the execution. Then among these k nodes, there should exist at least k-n/20 nodes who decide (including those who decided and then crashed) on the same value.**" Intuitively, here we only require k-n/20 nodes to agree with each other. The other standard requirements (i.e., **Termination** and **Validity**) for distributed consensus remain unchanged as before.

Do you feel you can design a deterministic algorithm for solving the distributed consensus problem under the above setting with the **Weakened-Agreement** requirement? (Yes/No) _____

If you indicated "Yes", please give an algorithm for solving the problem. Your answer should have 3 parts i) a high-level intuitive description of your central idea, ii) precise pseudo-code for your algorithm, and iii) a rigorous correctness proof. You are allowed to invoke any algorithm learned in class if you want, but you should clearly indicate which algorithm you invoke, and you should invoke the algorithm as a black-box. If you indicated "No", please give a rigorous impossibility proof. Only answering Yes/No without further details will not earn you any marks.

Problem 12 (6 marks)
Consider the distributed consensus problem where the timing model is synchronous, the channels are reliable, and the nodes may experience byzantine failures. Let p=0.01 and let q be any given constant where $0 < q < 1$. There are total n nodes, with ids from 1 through n. Each node will experience byzantine failure with probability p independently. Hence the expected number of nodes experiencing byzantine failures will be p*n. Given p and q, you are allowed to choose some arbitrary constant value m that is a function of p and q. Note that p, q, and m are all constants, and they do not change with n. You are asked to design a deterministic algorithm to solve the distributed consensus problem in the above setting, with the following requirements:

1. The value of n will be given to your algorithm. You can assume that **n will never be smaller than m**.
2. There should exist some function R(p,q,n) with the following property. Given p, q, and n, in your algorithm all nodes that do not experience byzantine failures should always decide within R(p,q,n) rounds, regardless of how many nodes actually experienced byzantine failures during the execution.
3. Your algorithm should minimize the asymptotic form of R(p,q,n) as n goes towards infinity.
4. With probability at least 1-q, your algorithm should satisfy the standard **Agreement** and **Validity** requirements in distributed consensus under byzantine failures.

You may choose to either use or not use the following fact: Let X be the sum of a sequence of independent binary random variables, where each binary random variable is 1 with probability p (and 0 with probability 1-p) independently. Let E[X] be the expectation of X. Then for any c>6, we have $Pr[X \geq c*E[x]] \leq 2^{(-c*E[x])}$.

Problem continues on the next page. **Write your answer on the next page...DO NOT write on this page.**

Problem 12 continues below:

What is the R(p,q,n), in its asymptotic form, of your algorithm? **O(** _____ **)** If you feel that the problem is not solvable under certain q, please write down infinity in the above bracket.

Please give your algorithm. Your answer should have 3 parts: i) a high-level intuitive description of your central idea, ii) precise pseudo-code for your algorithm, iii) a rigorous proof showing that your algorithm satisfies all requirements above. You are allowed to invoke any algorithm learned in class if you want, but you should clearly indicate which algorithm you invoke, and you should invoke the algorithm as a black-box. If you indicated "infinity" earlier, please give a rigorous impossibility proof.

END-OF-PAPER