Note: YOU MUST DO THE HOMEWORK BY YOURSELF. If you have difficulties in solving a question you may discuss it with friends, BUT you MUST phrase, write and formulate the answers by yourself, after you understand the solution. The language of the solution must be entirely your own.

Notice 2: Unless your hand writing is extremely clear, you should type (Word, or LaTeX, etc.) and print your homework.

Notice 3: Put your name and id number on each page of the solution.

1. The roman senate strikes again, this time they want to reach agreement (consensus) under the same conditions given in the previous homework: They want to agree on the binary input of one of them.

The senators gather in a ring around the roman forum, and the announcer declares the method (distributed algorithm) by which they will reach consensus. Senators then converse, each with his two neighbors on the ring, according to the announced method (distributed algorithm).

However, each senator might prefer that either 0 or 1 be decided, and thus may cheat in an attempt to increase the probability of its preferred value being decided. You have to devise a consensus algorithm that the announcer will declare to the senators, such that, no senator will gain anything from deviating (cheating) from your algorithm. Your algorithm should satisfy both the agreement and the validity conditions. The cheating is constrained as follows:

(a) If the algorithm fails to reach agreement, the senators are sent to prison for the rest of their life. Meaning, senators will not cheat if the cheating will definitely cause the algorithm failure. However, senators do take chances, if there is some positive probability to succeed.

(b) A senator cheats only if the cheating increases the probability of its preferred value being decided.

(c) Cheating senators assume they are the only cheaters.

Further make the following assumptions: (i) each senator has a unique name (known only to itself), (ii) each knows $n$, the total number of senators (iii) the senators converse in synchronous rounds (i.e., assume a synchronous ring), and (iv) assume all senators start together at the same round.

For any algorithm that you provide give its message and time complexities.

(a) Provide an algorithm when each senator prefers a certain value (0 or 1) not necessarily its own, and a subset of the senators may cooperate in cheating, to
increase the probability of a certain value. Each subset assumes that it is the only one to cheat. What is the largest size sub-set under which your algorithm works correctly? It should be the maximum size for which there is a solution. Notice, members of any subset may communicate only through the ring.

(b) Repeat the above questions for an synchronous complete network.

2. Is it possible to elect a leader in a synchronous ring with \( n \) nodes, \( n \) known to the nodes, node ID's are unique (no two ID's are the same), but the only operation the processors can do on the ID's is equal or not_equal comparisons. Provide a complete proof of your answer.

3. Use the DFS algorithm of the previous question (asynchronous, Bidirectional network, with unique id to each node) to derive an \( O(|E| + n \log n) \) messages election algorithm. HINT: Each initiator of the algorithm would start a DFS traversal. The question is how would colliding DFS traversals efficiently eliminate one another until only one DFS is left, which then captures the entire network and is elected as the leader.

4. Processors in a shared memory concurrent system are equipped with the operation Memory-Memory-Copy which atomically copies one shared memory word to another shared memory word. I.e., it is given two addresses, \( a1 \) and \( a2 \), and it atomically copies the content of Memory\[a1\] into Memory\[a2\]:

\[
\text{MM-copy}(a1, a2):
\text{Memory}[a2] := \text{Memory}[a1];
\]

What is the consensus number of this operation. That is, what is the maximum number of processors in such a system that can solve consensus when the only operations they may perform in the shared memory are atomic read or write and MM-copy\((a1, a2)\).

5. Describe a leader election algorithm in a shared memory concurrent system that supports atomic read and write operations and atomic binary-consensus. Operations on an atomic BConsensus\[x\] is invoked by any processor by a propose.BConsensus\[x\](input: boolean) operation. In its sequential specification the operation returns the input of the first processor that performed propose on BConsensus\[x\]. The LE, leader election algorithm returns the id of a processor that has invoked the elect\(\text{my_id}\) operation, to all the processors, i.e., it is like a multi valued consensus. Assume each processor id is represented by a vector of \(3 \log n\) bits.

6. (20 \%) In class we saw algorithm 15.5 (see Figure 1) ”A Wait-free universal algorithm using consensus objects”. (Recall the algorithm has a built in ”helping” mechanism so any processor finishes within a bounded number of iterations.)

(a) In each of the following changes, is the algorithm still correct? If we replace line 5 with:

i. priority := Head\[i\].seq mod n /* we took away the +1
ii. \( \text{priority} := \text{Head}[i].\text{seq} + 1 \mod 2n \)

iii. \( \text{priority} := 2 \cdot \text{Head}[i].\text{seq} + 1 \mod n \)

Please explain each of your answers. Why is it correct or what is wrong, and why.

(b) How many iterations of the ”while” loop (Lines 4–13) a processor may go through in the worst case in each of the three variants above.

Initially \( \text{Head}[j] \) and \( \text{Announce}[j] \) point to the anchor record, for all \( j, 0 \leq j \leq n - 1 \).

---

Initially \( \text{Head}[j] \) and \( \text{Announce}[j] \) point to the anchor record, for all \( j, 0 \leq j \leq n - 1 \).

When \text{inv} occurs:

1. allocate a new \text{opr} record pointed to by \( \text{Announce}[i] \) with \( \text{Announce}[i].\text{inv} := \text{inv} \) and \( \text{Announce}[i].\text{seq} := 0 \)

2. for \( j := 0 \) to \( n - 1 \) do

3. if \( \text{Head}[j].\text{seq} > \text{Head}[i].\text{seq} \) then \( \text{Head}[i] := \text{Head}[j] \)

4. while \( \text{Announce}[i].\text{seq} == 0 \) do

5. \( \text{priority} := \text{Head}[i].\text{seq} + 1 \mod n \)

6. if \( \text{Announce}[\text{priority}].\text{seq} == 0 \) is help needed

7. then \( \text{point} := \text{Announce}[\text{priority}] \)

8. else \( \text{point} := \text{Announce}[i] \)

9. \( \text{win} := \text{decide}(\text{Head}[i].\text{after}, \text{point}) \) try to thread chosen record

10. \( \text{win}.\text{seq} := \text{Head}[i].\text{seq} + 1 \)

11. \( \text{win}.\text{new-state}, \text{win}.\text{response} := \text{apply}(\text{win}.\text{inv}, \text{Head}[i].\text{new-state}) \)

12. \( \text{Head}[i] := \text{win} \)

13: enable the output indicated by \( \text{win}.\text{response} \)

---

Figure 1: The wait-free universal algorithm using consensus objects. Code for processor \( p_i \), \( 0 \leq i \leq n - 1 \)