Examen terminal

In the following we are working in the shared memory model with registers. By default (except in exercice 2 and in exercice 5 3.b) by implementation we mean wait-free implementation and every implementation may use an arbitrary number of registers.

Exercice 1.—

1. Depict a history of a one-writer one-reader register that satisfies the specification of a regular register, but does not satisfy the specification of an atomic register.

2. Is this a history of a regular register (Yes/No)? Why?

3. Is the history below linearizable with respect to the specification of queue? (Yes/No) If yes, assign a linearization point to each operation.

Exercice 2.— Is it possible to solve 3 set agreement if no more than 2 processes may crash? If yes give an algorithm.

Exercice 3.— Consider the class of objects Stb with the following specification:
• states: Objects in $\text{Stb}$ have 3 possible states: 0, 1 and $\perp$. Initially the state is $\perp$.
• write: A write does not return anything. A write($v$) modifies the state of the object to $v$ if the state of the object was $\perp$ and has no effect if the state was different from $\perp$.
• read: A read() does not modify the state of the object and returns the current state of the object.

1. Show that an object of $\text{Stb}$ can solve binary consensus for an arbitrary number of processes?
2. Give an algorithm with $\text{Stb}$ objects to solve the consensus on at most $m$ input values using at most $\log_2(m)$ $\text{Stb}$ objects. Argue about an extension to an arbitrary $m$. Prove that, with binary consensus objects, consensus among an arbitrary number of values is solvable.

Exercice 4.— The move($a, b$) operation atomically copies the value of a register $b$ into a register $a$. Consider the following algorithm:

```plaintext
pref is an array [1..n] of values
Tab is an array [1..n][0..1] of values in {0, 1}
initially $\forall i : Tab[i, 0] = 1$ and $\forall i : Tab[i, 1] = 0$

code for process $p \in \{1, ..., n\}$:
1. $\text{pref}[p] := \text{input}$
2. move($\text{Tab}[p, 1], \text{Tab}[p, 0]$)
3. for $i$ in $p+1...n$ do
4.   $\text{Tab}[i, 0] := 0$
5. for $i$ in $n..1$ do
6.   if $\text{Tab}[i, 1] == 1$ then
7.     decide $\text{pref}[i]$
```

1. Prove that this algorithm solves consensus among $n$ processes.
2. Replace line 2 of the algorithm by:
   $\text{Tab}[p, 1] := \text{Tab}[p, 0]$.
   Does this algorithm solves consensus among $n$ processes? If not give an execution in which the agreement property is not ensured.
3. Is the move($a, b$) shared by 2 processes implementable with FIFO? Is the move($a, b$) object shared by more than 2 processes implementable with FIFO objects?

Exercice 5.— In the following, a (one-shot\(^1\)) immediate snapshot is an object whose the state is a set of process indexes initially empty with a WriteSnap() operation returning a set of process indexes corresponding to a state of the object. More precisely, let $V_i$ be the set of process indexes returned by a call of WriteSnap() by process $i$ then the following properties are ensured:

- Termination: a WriteSnap() invocation made by a correct process terminates (i.e. a process making an infinite number of steps).
- Self-inclusion: $\forall i : i \in V_i$
- Validity: $\forall i : j \in V_i \Rightarrow j$ invoked WriteSnap
- Containment: $\forall i,j : (V_i \subseteq V_j) \lor (V_j \subseteq V_i)$

\(^1\)One-shot means that any process may invoke at least one time the operation (here the WriteSnap)
• Immediacy: \( \forall i, j : i \in V_j \Rightarrow V_i \subseteq V_j \)

1. Prove that immediate snapshot as defined before is not linearizable: give an execution satisfying the specification that is not linearizable.

If simultaneity (i.e. some operation may share the same linearization point) is allowed, does the immediate snapshot become “linearizable” (for this extension of the linearizability)?

2. Assuming a linearizable implementation of immediate snapshot (i.e. an implementation that is linearizable and satisfies the specification) proves that 2-consensus (consensus among 2 processes) is solvable with this implementation.

3. \( k \)-immediate snapshot is an immediate snapshot such that each set \( V_i \) contains at least \( n - k + 1 \) elements. Prove that:

   (a) with \( k \)-immediate snapshot \( k \)-set agreement is solvable. For which value of \( k \), \( k \)-immediate is implementable?

   (b) Assuming \( n > 2 \), with \( 2 \)-immediate snapshot and at most one crash consensus is solvable.