Exercise Session
Consensus

Exercise 1

Consider all our fail-stop consensus algorithms (Consensus Algorithm I and Consensus Algorithm II). Explain why none of those algorithms would be correct if the failure detector turns out not to be perfect.

A violation of *strong completeness* property of the perfect failure detector could lead to the violation of the *termination* property of consensus as follows. In all our fail-stop algorithms, there is at least one critical point where a process p waits to deliver a message from a process q or to detect the crash of process q. Should q crash and p never detect the crash of q, p would remain blocked forever and never decide.

Consider now *strong accuracy*. If it does not hold, our fail-stop consensus algorithms could violate the *agreement* property. It is easy to devise an execution where processes falsely suspect each other and hence decide on different values, thus violating *agreement*.

Exercise 2

Explain why any fail-noisy consensus algorithm (one that uses a $\diamond P$ failure detector) actually solves uniform consensus (and not only the non-univorm variant).

Consider any fail-noisy consensus algorithm that implements consensus but not uniform consensus. This means that there is an execution where two processes p and q decide differently and one of them crashes, so that the algorithm violates *uniform agreement*. Assume that process p crashes. With an eventually perfect failure detector, it might be the case that p has not crashed but is falsely suspected to have crashed by all other processes. Process q would decide the same as in the previous execution, and the algorithm would even violate the regular *agreement* property.

Exercise 3

Explain why any fail-noisy consensus algorithm (one that uses a $\diamond P$ failure detector) requires a majority of the correct processes. More precisely, provide a "bad run" in the case where there isn't a majority correct.

We explain this for the case of a system of four processes p, q, r, and s. Assume by contradiction that there is a fail-noisy consensus algorithm that tolerates the crash of two processes. Assume that p and q propose a value v, whereas r and s propose a different value u. Consider an execution E1 where p and q crash initially: in this execution, r and s decide u to respect the *validity* property of consensus. Consider also an execution E2 where r and s crash initially: in this scenario, p and q decide v. With an eventually perfect failure detector, a third execution E3 is possible: the one where no process crashes, p and q falsely suspect r and s, and r and s falsely suspect p and q. In this execution E3, processes r and s decide u, just as in execution E1 (they execute the same steps as in E1, and cannot distinguish E3 from E1 up to the decision point), whereas p and q decide v, just as in execution E2 (they execute the same steps as in E2, and cannot distinguish E3 from E2 up to the decision point). *Agreement* would hence be violated.