# Exercise 3: Impossible!

## Task 1: Stop Failing, You Cowards!

The goal of this exercise is to show that under the synchronous message passing model, for any consensus algorithm there are executions with $f$ crashes in which solving consensus requires at least $f+1$ rounds.

a) Show that there are inputs differing at a single *pivotal* node $v_0$ that result in different outputs in the respective (unique) maximal fault-free extensions.

   **Hint:** Use the same argument as for the asynchronous case.

b) Prove that, given a pair of $r$-round executions with a pivotal node $v_r$ (i.e., only this node's state makes the difference between outputs 0 and 1 in case there are no further faults), crashing the node "in the right way" yields a pair of $(r+1)$-round executions with a new pivotal node $v_{r+1}$.

   **Hint:** The reasoning is similar as for a), but the "inputs" are replaced by the messages of $v_r$ in round $r$ of each of the executions — or their absence due to the node crashing.

c) Conclude that for any $f < n$, there are executions with $f$ faults in which some node neither crashes nor terminates earlier than round $f+1$.

## Task 2: Impossible? We'll Do it in $f+2$ Rounds!

The topology: complete. The model: synchronous message passing. The task: consensus. The challenge: crash faults.

a) Suppose each node maintains a bit $p_i$. In each round, each node sends its bit to all other nodes and sets it to 0 if it received a 0. Show that if a node receives messages from the same set of senders either all with opinion 0 or all with opinion 1 in two consecutive rounds, all nodes have the same bit $p_i$.

b) Use this observation to construct a synchronous consensus algorithm tolerating an arbitrary number of faults.

c) Prove that the algorithm is correct and terminates in at most $f+3$ rounds in executions with at most $f$ faults (if necessary, modify your algorithm to achieve this property).

d)* Modify the algorithm to terminate in $f+2$ rounds under the assumption that $n$ is known!

Remark: Note that the algorithm can deal with an arbitrary number of faults, yet the running time is bounded in terms of the *actual* faults happening. This property is called *early-stopping*. Given that faults are supposed to be uncommon events, that's pretty neat!

## Task 3*: Intense Sharing

a) Find out what the term "consensus number" refers to!

b) Ponder the consensus number of shared memory that, besides atomic reads, permits to write to up to $k > 1$ shared registers in a single atomic step!

c) Share your insights in the exercise session!