Exercise 1 - NBAC & Weak Termination

Devise an algorithm that, without consensus, implements a weaker specification of NBAC by replacing the termination property with

**Weak termination**: Let $p$ be a distinguished process, known to all other processes. If $p$ does not crash then all correct processes eventually decide.

Your algorithm may use a perfect failure detector.
Solution 1

- Every process sends its proposal (COMMIT / ABORT) to $p$ using point-to-point links.
- $p$ collects all the proposals. If it detects (with the perfect failure detector) that any process crashed, or any process proposes ABORT then it unilaterally decides to ABORT. Otherwise, it unilaterally decides to COMMIT.
- $p$ uses Best-Effort Broadcast to send its decision to every other process. If $p$ does not crash, every correct process eventually receives $p$’s decision and decides accordingly.
Exercise 2 - NBAC & Very Weak Termination

Devise an algorithm that, without consensus, implements a weaker specification of NBAC by replacing the termination property with

**Very weak termination**: If no process crashes, then all processes decide.

Is a failure detector needed to implement this algorithm?
Solution 2

- Every process simply uses Best-Effort Broadcast to send its proposal to every other process.
- Upon receiving all proposals, a process decides COMMIT if it only received COMMIT proposals. It decides ABORT otherwise.
- Under the assumption that no process crashes, every process eventually receives the proposal of every other process, and decides.
- No failure detector was needed. Indeed, termination is not guaranteed if any process crashes.
Exercise 3 - TRB & ◇P

Can we implement TRB with an eventually perfect failure detector ◇P, under the assumption that at least one process can crash?
Solution 3

We cannot implement TRB with an eventually perfect failure detector. Let $s$ be the designated sender (broadcasting a message $m$), let $p$ be a correct process. Let us consider two executions, $A$ and $B$.

- In $A$, $s$ crashes before sending out any message. At time $t < \infty$, $p$ delivers ⊥.
- In $B$, $s$ is correct but all of its messages are delayed until $t' > t$. Moreover, ◇$P$ behaves identically in $A$ and $B$ until time $t$. This is possible because ◇$P$ is only eventually perfect.

Since $A$ and $B$ are indistinguishable, $p$ delivers ⊥ in $B$ as well. By agreement, $s$ delivers ⊥ in $B$. But this violates validity: $s$ should deliver $m$ in $B$. 
Exercise 4 - TRB to Consensus

Design an algorithm that implements consensus using multiple TRB instances.
Solution 4

- Every process uses TRB to broadcast its proposal.
- Let $p$ be any process, eventually every correct process either delivers $p$’s proposal or $\perp$ (if $p$ fails).
- Eventually, every correct process has the same set of proposals (at least one is not $\perp$, since not every process crashes).
- Processes use a shared but arbitrary function to extract a decision out of the set of proposals (e.g., sort alphabetically and pick the first).
Exercise 5 - TRB to Total Order Broadcast

Design an algorithm that implements Total Order Broadcast using multiple TRB instances.
Solution 5

We have already proven that we can implement Total Order Broadcast using multiple rounds of consensus. In the previous exercise, we proved that we can implement consensus using Terminating Reliable Broadcast.