Problem 1 You may recall that when we discussed the protocol for solving terminating reliable broadcast in a synchronous system with benign failures, we mentioned that it is possible to come up with an early-stopping version of the protocol, one in which the protocol stops in a number of round that depends on $t$, the actual number of failures experienced during the execution of the protocol, rather than $f$, the maximum number of failures that the protocol can tolerate.

Design an early stopping TRB protocol for benign failures, and prove it correct.

Problem 2 Suppose that we have an algorithm that solves consensus in $f + 1$ rounds in a synchronous system and tolerates $f$ crash failures. Use that consensus algorithm to solve Terminating Reliable Broadcast for crash failures in $f + 2$ rounds in a synchronous system with $f$ crash failures. You have to present your algorithm and prove that it satisfies Termination, Agreement, Validity and Integrity.

Problem 3 Under the same assumptions of the previous problem, use the consensus algorithm to solve Terminating Reliable Broadcast for crash failures in $f + 1$ rounds in a synchronous system with $f$ crash failures. (This is a challenging one!) As before, you have to present your algorithm and prove that it satisfies Termination, Agreement, Validity and Integrity.

Problem 4 Informally, a History system $H$ behaves as follows: on receiving a client request, the system returns the sequence of requests received until then from all clients. Clearly, the semantics of $H$ are extremely useful as any other deterministic system can be implemented using $H$.

In $H$ a client $c$ can initiate a request by calling the function REQUEST($c$, $op$), where $op$ is the next request (assume, for simplicity, that the argument list to the request is empty).

Consider the following implementation of a History system using a single non-faulty server $s$. Here is the function REQUEST:

```plaintext
function REQUEST($c$, $op$)
    send ($c$, $op$) to server $s$
    when received $res$ from $s$:
        return $res$
```

And here is the protocol executed by server $s$:

```plaintext
do forever
    when received ($c$, $op$)
        $sequence := sequence \circ op$
        send $sequence$ to client $c$
```
The above implementation guarantees that if a correct client initiates a request, then the client will eventually receive a response since \( s \) is non-faulty. We now require every History system \( \mathcal{H} \) to "behave" like the one implemented on this non-faulty server \( s \). More formally, let the run of any History system consist of two kinds of events—send\((c, op, t)\) if client \( c \) initiates \textbf{REQUEST}(\( c, op \)) at time \( t \), and recv\((c, op, res, t)\) if client \( c \) receives a response \( res \) at time \( t \) to the above request.

We assume that a client initiates a new request only after the response to the earlier request (if any) has been received. Let \( \omega_{\mathcal{H}_s} \) be the set of all infinite runs of an History system implemented on \( s \), and let \( \omega_{\mathcal{H}} \) be the set of infinite runs of any other History system \( \mathcal{H} \). Then our criterion for the correctness of \( \mathcal{H} \) is:

**Consistency** The History system \( \mathcal{H} \) is equivalent to the one implemented on the non-faulty server \( s \), i.e. \( \omega_{\mathcal{H}} \subseteq \omega_{\mathcal{H}_s} \).

Assume that between any two processes there is a reliable FIFO channel.

1. Show that the History system \( \mathcal{H} \) can be used to solve Consensus. (Hint: Assume that \( \mathcal{H} \) can be constructed using \( n_s \geq 1 \) servers. Show how to solve consensus using the \( n = n_s + 2 \) process system consisting of the \( n_s \) servers and 2 clients (say \( c_i \) and \( c_j \)) that tolerates a single crash failure).

2. What conclusion can be drawn from your proof of Part 1?

**Problem 5** Prove that the protocol discussed in class that uses a \textit{strong} failure detector solves Consensus (see notes).