**Problem 1.** Fetch-and-increment has a consensus number of 2, while compare-and-swap (CAS) has an infinite consensus number. Therefore we will use the universal construction to implement a fetch-and-increment object from consensus objects. Then we can replace consensus objects with their implementation\(^1\) from CAS objects. The resulting algorithm is a wait-free implementation of fetch-and-increment from CAS.

Universal construction algorithm for fetch-and-increment: *Shared objects:*

- Array of \( n \) atomic registers \( R[1, \ldots, n] \), where \( n \) is the number of processes.
- Infinite list \( C \) of consensus objects.

*Local objects:*

- register \( seq \) the value of which is the number of executed operations by process \( p[i] \), initially \( seq = 0 \).
- register \( k \) the value of which is the number of decided batches of requests, initially \( k = 0 \).
- list \( Perf \) of performed requests.
- list \( Inv \) of requests which need to be performed.
- local copy \( f \) of fetch-and-increment.

Pseudocode for process \( p[i] \):

```
fetch&inc()
    seq ++
    R[i] := (fetch&inc(), i, seq) // inform other processes about the request
    repeat
        Inv := Inv + R[1, \ldots, n].read // add new requests of other processes to the list
        Inv := Inv - Perf // remove performed requests from the list
        if Inv \( \neq \emptyset \) then // if there are requests that were not performed
            k++
            Dec := C[k].propose(Inv) // decide on requests to be performed
            Res := f.Dec // perform all requests from Dec on local copy \( f \)
            // and record the responses to list \( Perf \)
            Perf := Perf + Dec // add the performed responses to list \( Perf \)
            if (fetch&inc(), i, seq) \( \in \) Dec then // if the request by \( p[i] \) is in
                // the list of decided responses
                return the result of (fetch&inc(), i, seq) from Res
            // return the corresponding response
```

\(^1\)For the implementation of consensus from CAS see the lecture on the limitations of registers.