Algorithm 5.4 k-set consensus algorithm in the presence of crash failures: code for processor p_i, $0 \le i \le n-1$.

```
Initially V = {x}

round k, 1 ≤ k ≤ f/k + 1:                          // assume that k divides f
    send V to all processors
    receive S_j from p_j, 0 ≤ j ≤ n − 1, j ≠ i
    V := V ∪ ⋃_{j=0}^{n−1} S_j
    if k = f + 1 then y := min(V)                  // decide
```

5.4 Design a consensus algorithm for crash failures with the following *early stopping* property: if f' processors fail in an execution, then the algorithm terminates within $O(f')$ rounds.

Hint: Processors need not decide in the same round.

5.5 Define the k*-set consensus* problem as follows. Each processor starts with some arbitrary integer value x_i, and should output an integer value y_i such that:

Validity: $y_i \in \{x_0, \ldots, x_{n-1}\}$, and

k***-agreement:*** the number of different output values is at most k.

Show that Algorithm 5.4 solves the k-set consensus problem in the presence of f crash failures, for any $f < n$. The algorithm is similar to Algorithm 5.1 (for consensus in the presence of crash failures) and is based on collecting information.

What is the message complexity of the algorithm?

5.6 Present a synchronous algorithm for solving the k-consensus problem in the presence of $f = n - 1$ crash failures using an algorithm for consensus as a black box. Using Algorithm 5.1 as the black box, the round complexity of the algorithm should be $(\frac{n}{k} + 1)$, and its message complexity should be $O(\frac{n^2}{k}|V|)$, where $|V|$ is the number of possible input values. For simplicity assume that k divides n.

5.7 Show that the validity condition for Byzantine failures, given in Section 5.2.2, is not equivalent to requiring that every nonfaulty decision be the input of some processor. In particular, design an algorithm that satisfies the validity condition of Section 5.2.2 but does not guarantee that every nonfaulty decision is the input of some processor.

Hint: Consider the exponential message and phase king algorithms when the size of the input set is larger than 2.