Exercise #6
Due: February 26, 2008

6. Consider the synchronous message-passing model with a complete network graph. Up to f of the n processes may have Byzantine failures. We saw a consensus algorithm in class that satisfies
agreement: all correct processes produce same output, and
weak validity: if all correct processes have input v, they all output v.
This algorithm works regardless of the set of possible input values, as long as $n > 4f$.

A stronger validity condition is
strong validity: the output of every correct process is the input of some correct process.
Note that strong validity is equivalent to weak validity if the set of possible inputs is $\{0, 1\}$, but the conditions are not equivalent in general.

(a) Show that the algorithm from class does *not* satisfy strong validity if the set of possible inputs is $\{0, 1, 2\}$, even when $n > 4f$.

(b) Show that it is impossible to design an algorithm that satisfies termination, agreement and strong validity if $m = 5, n = 13$ and $f = 3$.

(c) Consider the problem of designing a consensus algorithm that satisfies agreement and *strong* validity. The domain of possible input values is $\{0, 1, 2, \ldots, m - 1\}$.

Show that the following algorithm satisfies agreement and strong validity when n is sufficiently large, relative to f and m. State clearly how big you are assuming n to be. (Of course, the weaker your assumption, the better.) Without loss of generality, you can assume that Byzantine processes always send a value when they are supposed to, but they may not send the right value. (If a process does not send a value to you when it is supposed to, you can just pretend it sent you 0).

Code for process i:

pref $\leftarrow$ input value
for phase $\leftarrow 1..f + 1$
    round 1:
    send pref to all processes (including self)
    suppose you receive k_j copies of v_j in this round, where $k_1 \geq k_2 \geq \cdots \geq k_m$.
    round 2:
    if phase $= i$ then send v_1 to all processes (including self)
    suppose the value received in this round is v_c
    if $k_1 - k_2 > 2f$ then pref $\leftarrow v_1$
    elsif $k_c > f$ then pref $\leftarrow v_c$
end for
output pref