1. Consider a system of *unbounded clocks* ticking at the same rate, and displaying the same value. Due to electrical disturbances, the phases of these clocks might occasionally be perturbed. The following program claims to synchronize their phases in a bounded number of steps following a perturbation:

```plaintext
{program for clock i}
define c : array [0 .. n-1] of integer {c[i] the value of clock i}
    {N(i) denotes the set of neighbors of clock i}
do
    true Æ c[i] := 1 + max {c[j] : j Œ N(i) \ {i}}
od
```

Assuming a synchronous model where all clocks execute the step with each tick, and the action requires zero time to complete, verify if the claim is correct. Prove using a well-founded set and an appropriate variant function that the clocks will be synchronized in a bounded number of steps. Also, what is the round complexity of the algorithm?

2. If a distributed computation does not terminate with a strongly fair scheduler, than can it terminate with a weakly fair scheduler? What about the converse? Provide justification (or example) in support of your answer.

3. In a tree with undirected edges, each node represents a process that has a color 0 or 1. Starting from an arbitrary initial configuration, our goal is to color the nodes in such a way that no two neighboring nodes have the same color. We propose the following algorithm for each process $i$: 
program twocolor
define c[i]: color of process i \{c = 0 or 1\}

do j: j \in N(i) :: c[i] = c[j] \[ c[i] := 1 - c[i] \] od

Assume that the scheduler is weakly fair. Will the algorithm terminate? If not, then explain why not. Otherwise, give a proof of termination.

4. Consider an array of \( n \) (\( n > 3 \)) processes. Starting from a one end of the array, mark the processes as even and odd. Assume that the even processes have states \( \{0, 2\} \), and the odd processes have states \( \{1, 3\} \). The system uses the state-reading model, coarse grain atomicity, and distributed scheduling of actions. From an unknown starting state, each process executes the following program:

program alternator \{for process i\}
define s \in \{0, 1, 2, 3\}. \{state of a process\}
do j: j \in N(i) :: s[j] = s[i] + 1 mod 4 \[ s[i] := s[i] + 2 mod 4 \] od

Observe and summarize the steady state behavior of the above system of processes. What is the maximum number of processes that can eventually execute their actions concurrently?

5. The following reactive computation runs on a unidirectional ring of \( N \) processes 0, 1, 2, \ldots N-1. Processes 0 and N-1 are neighbors. Each process \( j \) has a local integer variable \( x[j] \) whose value is in the range 0..\( K-1 \) (\( K > N \)).

\( \{\text{process 0}\} \) do \( x[0] = x[N-1] \[ x[0] := x[0] + 1 \mod N \] od

\( \{\text{process j > 0}\} \) do \( x[j] \neq x[j-1] \[ x[j] := x[j-1] \] od

Prove that the above computation will not deadlock.