Homework Exercise #8
Due: December 6, 2011

8. Consider an asynchronous shared-memory system with halting failures. In class we proved that the following algorithm solves 2-process consensus using a stack *Stack* that initially contains one element "winner" and two registers $R[1]$ and $R[2]$ that each initially contain $\perp$.

```
1  CONSENSUS(v)                                    ▷ Code for process i (i ∈ {1,2})
2      R[i] ← v                                    ▷ Write your input to your register
3      if Stack.pop() ="winner" then result ← R[i] ▷ Choose your own input value
4      else result ← R[3 − i]                      ▷ Choose other process's input value
5      output result
6  end CONSENSUS
```

This algorithm uses a stack that is initially non-empty. We want to show it is possible to solve consensus using stacks that are initially empty (plus registers). Prove that the following algorithm correctly solves consensus for two processes. It uses two stacks $Stack_1$ and $Stack_2$ (initially empty), a single-writer snapshot object *Snap* (whose components are initially $\perp$) and four registers $R_i[j]$ where $i, j \in \{1, 2\}$ (all initially $\perp$).

```
1  CONSENSUS(v)                                    ▷ Code for process i (i ∈ {1,2})
2      pref ← v
3      Stack_i.push("winner")                      ▷ Initialize your stack
4      Snap.update(i, "ready")                     ▷ Announce your stack is ready
5      v⃗ ← Snap.scan()                             ▷ Find out which stacks are ready
6      for j ← 1..2
7          if v⃗[j] ="ready" then
8              R_j[i] ← pref                       ▷ These 3 lines similar to alg above
9              if Stack_j.pop() ="winner" then pref ← R_j[i]
10             else pref ← R_j[3 − i]
11         end if
12     end for
13     output pref
14 end CONSENSUS
```

Here's the intuitive description of what the algorithm does. There are really two copies of the basic consensus algorithm embedded in it. One copy uses $Stack_1, R_1[1]$ and $R_1[2]$. The other copy uses $Stack_2, R_2[1]$ and $R_2[2]$. Each process participates in one copy or both copies (depending on which stacks are ready to use when the process performs its *scan*). If a process participates in both copies, it uses the output from the first copy as its input to the second copy.

9. Prove that a system with stacks and registers can solve asynchronous consensus among n processes if we know there will be at most one process that has a halting failure. Your algorithm should be very simple.