Algorithm 4.7 Fast mutual exclusion: code for processor p_i, $0 \leq i \leq n-1$.

Initially, *Fast-lock* and *Slow-lock* are 0, and
Want$[i]$ is false for all i, $0 \leq i \leq n-1$

```
⟨Entry⟩:
1:   Want[i] := true
2:   Fast-lock := i
3:   if Slow-lock ≠ 0 then
4:       Want[i] := false
5:       wait until Slow-lock = 0
6:       goto 1
7:   Slow-lock := i
8:   if Fast-lock ≠ i then                    // otherwise, enter on the fast path
9:       Want[i] := false
10:      for all j, wait until Want[j] = false
11:      if Slow-lock ≠ i then                // otherwise, enter on the slow path
12:          wait until Slow-lock = 0
13:          goto 1
⟨Critical Section⟩
⟨Exit⟩:
14:  Slow-lock := 0
15:  Want[i] := false
⟨Remainder⟩
```

A fast algorithm clearly requires the use of multi-writer shared variables; if each variable is written only by a single processor, then a processor wishing to enter the critical section has to check at least n variables for possible competition.

The key for the algorithm is the correct combination of two mechanisms, one for providing fast entry when only a single processor wants the critical section, and the other for providing deadlock freedom when there is contention.

In the algorithm presented below, two shared variables are used for controlling access when there is no contention: *Fast-lock* and *Slow-lock*. In addition, each processor p_i has a Boolean shared variable *Want*$[i]$ whose value is true if p_i is interested in entering the critical section and false otherwise. The pseudocode appears in Algorithm 4.7.

A processor can enter the critical section either by finding *Fast-lock* $= i$ (in Line 8) or by finding *Slow-lock* $= i$ (in Line 11). In the first case (Line 8) we say that a processor *enters the critical section along the fast path*, while in the second case (Line 11) we say it *enters the critical section along the slow path*.

Consider the simple case in which no processor is in the critical section or in the entry section; in this case, *Slow-lock* is 0 and all entries of *Want* are 0. When