it decrements the number of forks available to its neighbors. After eating, a philosopher calls `releaseForks`, which, in addition to updating the array `fork`, checks if freeing these forks makes it possible to signal a neighbor.

Let *eating* [*i*] be true if philosopher *i* is eating, that is, if she has successfully executed `takeForks (i)` and has not yet executed `releaseForks(i)`. We leave it as an exercise to show that $eating[i] \leftrightarrow (forks[i] = 2)$ is invariant. This formula expresses the requirement that a philosopher eats only if she has two forks.

Algorithm 7.5. Dining philosophers with a monitor

```
monitor ForkMonitor
  integer array[0..4] fork ← [2, . . . , 2]
  condition array[0..4] OKtoEat
  operation takeForks(integer i)
    if fork[i] ≠ 2
       waitC(OKtoEat[i])
    fork[i+1] ← fork[i+1] - 1
    fork[i-1] ← fork[i-1] - 1
  operation releaseForks(integer i)
    fork[i+1] ← fork[i+1] + 1
    fork[i-1] ← fork[i-1] + 1
    if fork[i+1] = 2
       signalC(OKtoEat[i+1])
    if fork[i-1] = 2
       signalC(OKtoEat[i-1])
```

philosopher i

```
    loop forever
p1:     think
p2:     takeForks(i)
p3:     eat
p4:     releaseForks(i)
```

7.5. Theorem

Algorithm 7.5 is free from deadlock.

Proof: Let *E* be the number of philosophers who are eating, In the exercises, we ask you to show that the following formulas are invariant:

7-1.

$$\neg\, empty(OKtoEat[i]) \rightarrow (fork[i] < 2).$$

7-2.

$$\sum_{i=0}^{4} fork[i] = 10 - 2 * E.$$

Deadlock implies $E = 0$ and all philosophers are enqueued on `OKtoEat`. If no