```
36    public void await() {
37      int me = ThreadInfo.getIndex();
38      Node myLeaf = leaf[me / radix];
39      myLeaf.await(me);
40    }
41    private class Node {
42      AtomicInteger count;
43      Node parent;
44      volatile boolean sense;
45      int d;
46      // construct root node
47      public Node() {
48        sense = false;
49        parent = null;
50        count = new AtomicInteger(radix);
51      }
52      public Node(Node myParent) {
53        this();
54        parent = myParent;
55      }
56      public void await(int me) {
57        boolean mySense = threadSense.get();
58        // visit parent first
59        if ((me % radix) == 0) {
60          if (parent != null) { // root?
61            parent.await(me / radix);
62          }
63        }
64        int position = count.getAndDecrement();
65        if (position == 1) {  // I'm last
66          count.set(radix);   // reset counter
67          sense = mySense;
68        } else {
69          while (sense != mySense) {};
70        }
71        threadSense.set(!mySense);
72      }
73    }
74  }
```

FIGURE 18.19 Reverse tree barrier part 2: correct or not?.