Bob the COMP 322 TA proposes a solution to the Dining Philosophers Problem that is based on the combination of our solutions 1 and 2, by making Philosopher 0 “special”. All the philosophers follow the solution 2, except philosopher 0, who does not use tryLock’s, but instead uses regular blocking locks. Bob’s algorithm is on the right.

Worksheet: Dining Philosophers

1. int numPhilosophers = 5;
2. int numChops = numPhilosophers;
3. Chop[] chop = ...; // Initialize array of chopsticks
4. for(p in 1..numPhilosophers-1) { // philosophers 1..5
5.   async() -> {
6.     while(true) {
7.       Think;
8.       if (!chop[p].lock.tryLock()) continue;
9.       if (!chop[p-1].lock.tryLock()) {
10.          chop[p].lock.unLock(); continue;
11.       }
12.     }
13.   } /async*/
14. } /for*/
15. async() -> { while(true) { // philosopher 0
16.     Think;
17.     chop[0].lock.lock(); chop[numChops].lock.lock();
18.     Eat;
19.     chop[0].lock.unlock(); chop[numChops].lock.unlock();
20. } /async*/