Worksheet: Dining Philosophers

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.

Does Bob's solution exhibit:

1. Deadlock NO
2. Livelock NO
3. Starvation YES
4. Non-concurrency YES
5. Fairness (i.e. all philosophers get an equal chance to eat) NO

```
1. int numPhilosophers = 5;
2. int numChops = numPhilosophers;
3. Chop[] chop = ... ; // Initialize array of chopsticks
4. for(p in 1 .. numPhilosophers-1) { // philosophers 1 .. 4
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.     Eat ;
13.     chop[p].lock.unlock();chop[p-1].lock.unlock();
14.   } /*while*/ }); /*async*/ } /*for*/
15. async(() ->{ while(true) { // philosopher 0
16.     Think ;
17.     chop[0].lock.lock(); chop[numChops-1].lock.lock();
18.     Eat ;
19.     chop[0].lock.unlock();chop[numChops-1].lock.unlock();
20.  } /*while*/ }); /*async*/
```