**Name: Martin L. Metke**

**Problem Number:** 1.D

Execution State:

5 Dining Philosopher threads share the dp monitor and utilize the Concurrent Pascal paradigm of completely exiting the monitor after calling signal(). In this case, only one call to test() from within the returnForks() method will complete.

Problem encountered:

This change engenders a higher chance for some number of processes to starve, or for some Dining Philosophers to starve, because they will never be signaled and will never check for their turn to eat. However, it does not \*guarantee\* starvation, due to the cyclical nature of the “table” and the fact that the check to the left still operates correctly. It is possible that the ability to eat will propagate clockwise.

|  |  |
| --- | --- |
| Process #0 – returnForks() thread | |
| Line # | 21 ( test((i + 4) % 5); ) |
| Relevant Variables | |
| self [ (i + 4) % 5] | The Dining Philosopher who is returning her forks notifies the (i + 4) % 5–th Dining Philosopher (her left-hand neighbor) that she is done with her fork on the left side and signals that neighbor, which causes the original Philosopher process to immediately exit from both the function test() and the function returnForks(). |

|  |  |
| --- | --- |
| Process #1 – takeForks() thread | |
| Line # | 15 ( self.wait(); ) |
| Relevant Variables | |
| self[ (i + 1) % 5 ] | This thread – the Dining Philosopher “to the right” of Process #0 – will not be awoken because the call to test(( i + 1 ) % 5); is skipped when Process #0 calls signal() and exits on the left-hand neighbor. There is no guarantee that this process will ever wake up, leading to starvation. |