Brandon Fowler

lab6prob1 output

```
Philosopher 1 is Thinking
Philosopher 1 is Hungry
Philosopher 3 is Thinking
Philosopher 4 is Thinking
Philosopher 4 is Hungry
Philospher 4 is Eating
Philosopher 4 is Thinking
Philosopher 4 is Hungry
Philospher 4 is Eating
Philosopher 3 is Hungry
Philosopher 0 is Thinking
Philospher 3 is Eating
Philosopher 0 is Hungry
Philosopher 2 is Thinking
Philospher 0 is Eating
Philosopher 4 is Thinking
Philosopher 0 is Thinking
Philosopher 4 is Hungry
Philospher 1 is Eating
Philospher 4 is Eating
Philosopher 2 is Hungry
Philospher 2 is Eating
Philosopher 2 is Thinking
Philosopher 2 is Hungry
Philospher 2 is Eating
Philosopher 2 is Thinking
Philosopher 2 is Hungry
Philospher 2 is Eating
Philosopher 2 is Thinking
Philosopher 2 is Hungry
Philospher 2 is Eating
Philosopher 2 is Thinking
Philosopher 2 is Hungry
Philospher 2 is Eating
Philosopher 2 is Thinking
Philosopher 0 is Hungry
```

lab6prob1 question

By looking at the output, I can see that each thread is being given its chance to run, and at a reasonable frequency so that starvation is avoided. Threads run in an order that is at the mercy of the scheduler. For example philosopher 1 becomes hungry first , but waits until 4, 3, and 0 have eaten before eating. Some philosophers are stopped by others eating, but dead lock does not occur; and two philosophers can eat at once, as shown above by 4 and 1 eating at the same time.

<u>Lab6prob2 output</u>

```
Reader 1 reading
Reader 1 reading
Reader 1 reading
Reader 1 reading
Reader 1 reading
Writer 0 locking db
Writer 0 writing
Writer 0 unlocking db
Reader 1 reading
Reader 1 reading
Reader 1 reading
Reader 0 reading
Writer 0 locking db
Writer 0 writing
Writer 0 unlocking db
Reader 2 reading
Reader 3 reading
Reader 1 reading
Reader 3 reading
Reader 4 reading
Reader 1 reading
Writer 1 locking db
Writer 1 writing
Writer 1 unlocking db
Reader 0 reading
Reader 0 reading
Reader 0 reading
Writer 2 locking db
Writer 2 writing
Writer 2 unlocking db
```

<u>lab6prob2 question</u>

By looking at the output I can see that all 5 readers and all 3 writers initialized in this test, are taking turns running based on the scheduler. Neither deadlock nor starvation occurs for any thread. Multiple readers can read “at once” as shown by 2, 3, 1, and 4 above. However, as each writer is allowed to run, both other writers and all readers are locked out until the current writer is finished. This is illustrated above in the fact that no other threads output while a writer preforms its atomic operation.