Evan Nichols
EECS 678
Lab 7 Report
4-5-16

1. The asymmetric solution forces even-numbered philosophers to pick up their chopsticks Left-then-Right, while odd-numbered philosophers pick theirs up Right-then-Left. This way, all philosophers acquire one chopstick and randomly compete for the other. As long as eating and thinking times are random, each philosopher should not starve. This prevents deadlock because initially one philosopher will successfully pick up both chopsticks, eat, and then release them again, which sets off a “chain reaction” of waiting philosophers now able to acquire their second chopstick.
2. The asymmetric solution does NOT prevent starvation. As the slides mention, if a set of philosophers ever began to share the same “rhythm,” one philosopher might be at risk of starvation. Consider two philosophers with nearly identical eat and wait times, and a third philosopher with an “off-pattern” eat and wait times. A situation could arise where the third philosopher is “locked out” because it is always thinking during times the necessary chopstick is available.
3. Using a waiter mutex, this solution prevents deadlock by using a condition variable. A thread acquires the waiter mutex. While the necessary chopsticks are NOT free, the thread waits (using a pthread_cond_wait() call). Once available, the chopsticks are marked as taken, the waiter mutex is unlocked, and the philosopher eats. After eating, the waiter mutex is acquired again, the chopsticks are marked as available, and the philosophers neighbors are signaled of the change (using pthread_cond_broadcast() call).
4. The waiter solution does NOT prevent starvation. As mentioned in the slides, there is a “small but significant percentage of time where a chopstick is taken between when the signal is sent and when the receiving philosopher tries to get its chopsticks. In this way, a philosopher could miss its window of opportunity to acquire the chopsticks.
5. The Phil may find that both of its chopsticks are NOT free if both of its neighbors acquired its freed chopsticks before sending that neighbor a signal. This could be caused by the interleaving of different thread instructions.