1. Semaphore bowlAvailable is used to enforce the rule that only one creature may eat out of one bowl. It only allows as many creatures as there are bowls into the critical section.

   Lock cat is used to enforce mutual exclusion in the cat synchronization code.

   Lock mouse is used to enforce mutual exclusion in the mouse synchronization code.

   Condition variable move is used to enforce turns for the cats and mice. When the one group of creatures finishes eating, the condition variable forces the other group to start eating. This way only one group may eat at one time. Move is used on the flag integer.

   int bowlCount keeps track of the next available bowl number.

   int eatCount keeps track of the number of creatures in the group that have already eaten.

   int mice and int cats counts the number of cats and mice that passed through the system. These are used to determine whether or not there are cats or mice in the system.

2. The condition variable along with the flag integer are used to synchronize the cats and mice. When one group is done eating, the flag is triggered, and the next group is allowed to eat. The condition variable forces separation for the two groups of creatures.

3. Two creatures cannot pick the same bowl at the same time because the only one group may eat at one time, as well as because the bowl numbers are chosen preemptively. A creature would get the next available bowl, and once all the bowls have filled up, semaphore bowlAvailable prevents any more creatures from entering the critical section, they must wait for bowls to become available.

4. Condition variable along with flag integer prevents cats and mice from eating at the same time. Only one group of creatures may eat at any time. The cats and mice take turns eating.

5. It's not possible for cats or mice to starve because the synchronization technique forces alternation. The cats and mice take turns eating, each creature being allowed to eat each turn.

6. There is no bias in my implementation. It is both fair and efficient. It is fair because the groups alternate and take turns eating. Also, every single creature in the group gets an opportunity to eat when it's the groups turn. It is fairly efficient because most bowls are used. I believe my solution is balanced in turns of efficiency and fairness due to the time sharing principle I implemented. Every creature gets the same opportunity to get the resources, and has to wait the same amount of time between meals as every other creature.