Question 1.

```
struct cv * cats;
struct cv * mouse;
volatile int numCatsEating = 0;
volatile int numMouseEating = 0
struct lock * catEatLock;
struct lock * mouseEatLock;
struct lock * bowlLock;
```

Question 2.

```
struct cv * cats;          //conditional variable to tell cats to wait while mouse is eating
struct cv * mouse;         //conditional variable to tell mouse to wait while cat is eating
volatile int numCatsEating = 0;   //integer to keep track of the number of times cats eating
volatile int numMouseEating = 0; //integer to keep track of the number of mouse eating
struct lock * catEatLock;          //lock to lock the thread when cat is eating
struct lock * mouseEatLock;        //lock to lock the thread when mouse is eating
struct lock * bowlLock;            //lock to lock the bowl when the bowl is in use
```

Question 3.

I create an array of Booleans that is the size of the number of bowls. Each time the bowl is in use, the index of the bowl is set the true. Each time the bowl is chosen, I create a guard such that a bowl that is being used cannot be chosen.

Question 4.

The conditional variables are created and locks the opposing thread when either cat or mouse is eating from the bowl. Then, when the first animal finishes eating, the cv broadcasts to the opposing thread allowing that opposing animal to eat.

Question 5.

The locks and cv's work hand in hand to aid the animal to complete eating. So… all of the cats will eat first, then mouse will eat, then cat, etc.

Question 6.

I sacrifice efficiency for fairness. When a cat starts to eat, then a mouse waits to eat, then a cat waits to eat, the cat will eat first. This is because I've programmed it for all of one species to eat first before the second species goes to eat.