Question1

I have used

1. **Conditional variables** *bowlStatusForCat* and *bowlStatusForMouse*.
   Each of them is used to control the behavior of cats and mice. Such as when to eat and when to wait by putting them in to wait channel. When one creature is done eating, it will broadcast to other creatures.
2. **Lock** *Animal_lock*,two animals share one lock to protect the shared variables.
3. **Shared Variables** *numberOfWaitingCat,numberOfWaitingMouse, bowlAvialable, currentAnimal,* and a *char Array bowls.*
   numberOfWaiting is used to keep track on how many animals out there.
   bowlAvialable is a integer used to keep track on how many empty bowls available.
   The Char array is a duplicate of the one in bowls.c, I use this array to keep track on bowl status. Each bowl will have '-','m',and 'c' three status, representing empty, mouse and cat, respectively.

Question 2

*Struct lock * Animal_lock* is the lock I used to protect the critical sections which contain my shared variables *INT numberOfWaitingCat/mouse, bowlsAvailable, and Char *bowls.* The lock ensures the critical sections are under protection. The eating behavior is not locked because more than one cat can eat together, but they have to be assigned a bowl one by one thus other shared variables are locked.

Question 3

I used an *Char array* to store the status of each bowl, initialized by '-' . Whenever a cat/ mouse entered the status of the first empty bowl available is changed to 'c'/'m' respectively. A Cat/Mouse will walk through the array to search for available bowl. If all the bowls are occupied, then the creature has to wait. Also the array is protected under lock, thus ensures no two creatures will eat same bowl.

Question 4

I used a shared variable called *Char currentAnimal*. This char variable represent which creature is eating right now, so the other have to wait. Also in the while loop at top, I will check if the entering animal is same as the currentAnimal, if it's not, we let the entering animal wait until it's their turn to eat.

Question 5

I am always tracking on numberOfCatWaiting/numberOfMouseWaiting. The integer is only decremented only if the creature finished all the loops. And if the waiting animal number is greater than zero the current animal will broadcast all the other animal if there is any. For example if the cats finished this round and there is still mice waiting, they will broadcast to the mice wait channel. However if there is no mouse waiting, they will keep eating. Note I used two CV here, one channel for mouse and one channel for cat. They should broadcast each other to eat until no one starves eventually.

Question 6

My design emphasizes fairness. Cat and mouse eat turn by turn. Because at the end of each loop, if there is no current creature is eating, then it will broadcast to the other creature's wait channel, therefore it is fair to the other creature. For example if cats eat first then the eating order should be cmcmcmcmcm ect. Note cat and mouse threads must compete for who eat first. The one who first enters get to eat first, therefore fairness is reached. I did balance efficiency and fairness in some degree. If the last cat is done eating and the mice is still sleeping, then the cats will still eating then.