Question 1.
Answer: I used:
An array of locks named locks that each lock saves the lock of each bowl.
A cv named noCats that indicates no cat eating the bowls.
A cv named noMice that indicates no mouse eating the bowls.
A shared int named countCats that indicates the number of the cat eating the bowls.
A shared int named countMice that indicates the number of the cat eating the bowls.

Question 2.
Answer: In my synchronization each bowl is represented by a lock and placed in an array called blocks. Then in mouse and cat simulation each cat(mouse) thread attempt to eat a bowl will be blocked if the bowl is acquired. And I use two condition variable noCats and noMice to:1. Broadcast there are no cats(mice) eating the bowl, then the other creature can begin to eat. 2. Block mice(cats) while cats(mice) are eating bowls. To use the condition variable I defined two shared ints named countCats and countMice as the condition to broadcast or block .

Question 3.
Answer: Since I set a lock for each bowl, then the first creature eating the bowl will acquire the lock, thus before it releasing the lock the other creature cannot acquire the lock hence cannot eat the bowl.

Question 4.
Answer: Since all the mice threads are waiting until the cv noCats broadcasts and cv noCats indicates there is no cat eating, that means the mice won't eat until the cats finish eating. Thus no mice would be eaten by the cats.

Question 5.
Answer: Since every cat and mouse will wait at the bowl which they acquired but failed, then once the bowl is released by other creature they can acquire again. Such process repeat many times base on the number of the cats and the mice and the looptimes until every creature eats looptimes. Thus no creature could starve at the end of the program.

Question 6.
Answer:
Fairness: Since I used the random generating function to decide which bowl should be eaten for each creature and the sleeping time for cat and mouse might be different, then the creature which has less sleeping time will have more chance to

finish eating as the number of bowls become larger. Such method is a little bit unfair for the creatures have more sleeping time.

Efficiency: Since I just use the random generating function to decide which bowl should be eaten for each creature, then there must be some unused bowl while many creatures are still waiting for the acquired bowl to eat. Such method is of course inefficient.

Summary: I think my method is almost fair for every creature when they have the same sleeping time and eating time, but since I let the creature who acquire first eat first, then the efficiency must be sacrificed.