Question#1:

1. First-come first serve (FCFS). Because beta is greater than alpha so once the waiting process can execute, it priority will increase by beta and no other process can preempt to execute.
2. Last In First Out. Because once the first previous one want to execute and it’s priority will decrease then the next one will execute.

Question#2:

I agree with it will favor I/O-bound process because it is the least CPU time in the recent past. So it always selects the least CPU time during ‘t’ then the I/O bound process will have more trend to be execute earlier.

I disagree with it will not permanently starve CPU-bound processes because the CPU-bound processes may be preempted by I/O-bound processes forever.

Question#3:

If time slide is not enough to execute those processes then they will be add to the end of the ready queue but next time when CPU runs that processes, the will processes will be run twice but also cannot be finished. The CPU will run as these way forever.

Question#4:

If there is only one chopstick beside him and the philosopher doesn’t have any chopsticks yet then he shouldn’t pick up the chopstick. What’s more, if no one has two chopsticks, he shouldn’t pick up. These rules will prevent deadlock.

Question#5:

A philosopher can allocate a chopstick by following rules:

1: A philosopher doesn’t have chopsticks and there have 2 chopsticks on the table and there at least has a philosopher has 2 chopsticks then this philosopher can allocate a chopstick.

2: A philosopher has a chopstick and there at least has 2 chopsticks.

3: A philosopher has 2 chopsticks and there at least has 1 chopstick.

4: There at least has 1 chopstick and at least 1 philosopher having getting 2 chopsticks.

Question#6:

The sum of all maximum needs is less than m+n. If there has a deadlock then the m resources have been seized by all n processes. So the remaining of the n processes really need should be (m+n) – m so the sum of n processes need is less than n which is conflict with the maximum need of each process is between 1 resource and m resources.