**2.** Assume the algorithm is not safe, both thread A and thread B buy milk. Considering those three cases below:

Case 1: (noteB = 1, milk = any value). When noteB = 1, thread A will enter the while loop and hanging there. Therefore, that is contrary to our assumption.

Case2: (note B = 0, milk>0). When milk >0, thread A will not buy milk. Therefore, that is contrary to our assumption.

Case3: (note B = 0, milk = 0). When note B is 0, that means thread B has already finished the process “leave note, check milk, decide if need to buy milk, remove milk.”. Therefore, it is impossible for thread B to buy milk when note B = 0. Thus, that is contrary to our assumption.

Since every case contradicts the assumption, the algorithm is safe.

**3.** case 1: thread workds[0] runs before testRemoval.

In that case, the program will try to insert 50 numbers from 0…49 to queues [0]. However, the max capacity of the Tsqueue is just 10 numbers. Therefore, after inserting number 9 into queues [0], the tryInsert function will do nothing but return false. After thread works [0] finished, the program will execute the for loop. Then it will output “Queue 0”. Then it starts removing items in the queues [0]. There is total 10 numbers in the queues [0], from 0…9. Therefore, the output will be “Removed 0, Removed 1, Removed 2, Removed 3, Removed 4, Removed 5, Removed 6, Removed 7, Removed 8, Removed 9.” Then queues [0] will be empty and tryRemove function will return false. Therefore, for the rest part of the for loop, the output will be “Nothing there.” × 10 times. Either thread for queue [1] and queue [2] has been executed yet, therefore, queue [1] and queue [2] has nothing inside, and their tryRemove functions will return false. Therefore, for the testRemoval function with queue [1] and queue [2] will just print “Nothing there.” ×10 times In conclusion the output will be:

Queue 0: Queue 1:

Removed 0 Nothing there Nothing there Nothing there

Removed 1 Nothing there Nothing there Nothing there

Removed 2 Nothing there Nothing there Nothing there

Removed 3 Nothing there Nothing there Nothing there

Removed 4 Nothing there Nothing there Nothing there

Removed 5 Nothing there Nothing there Nothing there

Removed 6 Nothing there Nothing there Nothing there

Removed 7 Nothing there Nothing there Nothing there

Removed 8 Nothing there Nothing there Nothing there

Removed 9 Nothing there Nothing there Nothing there

Queue 2:

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

**Case 2:** thread workds [0] runs after testRemoval.

Because thread workds [0] runs after testRemoval, when testRemoval executed, there is nothing in the queue [0], and for queue [1] and queue [2] will be the same. Therefore, the output should be:

Queue 0: Queue 1:

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Queue 2:

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

Nothing there Nothing there

**10.** Because if we use writeGo.Boradcast, it will notify all the writers that in the waiting list to be ready to write while writeGo.Singnal will only inform one write in the waiting list to be ready to write. Due to it supposed to be only one writer to write at a time, we use writeGo.Singnal instead of writeGo.Boradcast.