• Why is it that you might observe that tasks just added something to the list and find the list empty straight after? Conversely: why is that tasks just read something from the list and still find the list full to the brim right after?

The first situation happens because: the queue check of the second producer has already been done before the first producer calls the Enqueue and fills in the item.

The second condition takes place because: one of the producer Enqueue an item in the queue during the period between consumer Dequeue and queue check.

• Would it be possible that a tasks states in the above program that the queue is empty and full?

Yes, it could happen when the Queue is empty during the Queue.Is\_Empty instruction, and then filled up with items by producers during the Queue.Is\_Full (the next instruction).

• Will all producers write the same number of elements?

Yes, because even though the queue might be full sometime, they will wait until they are able to add next item in the queue.

• Will all consumers read out the same number of elements?

No, because, in the concurrent environment, there is no guarantee that consumers can pick up elements in turn. Usually, they will Dequeue in random order and terminate when get an ‘f’.

• Will the total number of elements written equal the total number of elements read?

Yes, because the element added into the queue will finally be picked up by one of the consumer, and eventually leave the queue empty.

• Did we get rid of all exceptions for good? Did we introduced a new, potentially troublesome issue by our “hold instead of break” method?

No, there might still be some exceptions. If the producer keep holding state for long time, it will cause some conditions like freezing. What’s more, there is no guarantee as well that the interference cannot happen during the Enqueue and Dequeue, which leads to overflow and underflow.