Producer: LOW PRIORITY task
Consumer: HIGH PRIORITY task

```
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 1
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 0
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 0
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 1
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 0
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 0
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 1
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
            line 70, Switch value: 0
consumer(), line 67, rx
 onsumer(), line 67,producer(), line 50, tx
```

Explanation:

1. When the Producer task fills the queue and prints line 50.
2. The xQueueSend() will internally switch context to Consumer task because it is higher priority than the Producer task.
3. The Consumer task prints the switch vale at line 70, later prints line 67 as continues in its while loop.
4. Then, when the xQueueReceive() sleeps for portMAX_DELAY
5. The Producer task resumes out of xQueueSend()and go over to the next line 56
6. But as Producer task has vTaskDelay of 300ms the context switches back to Consumer task.

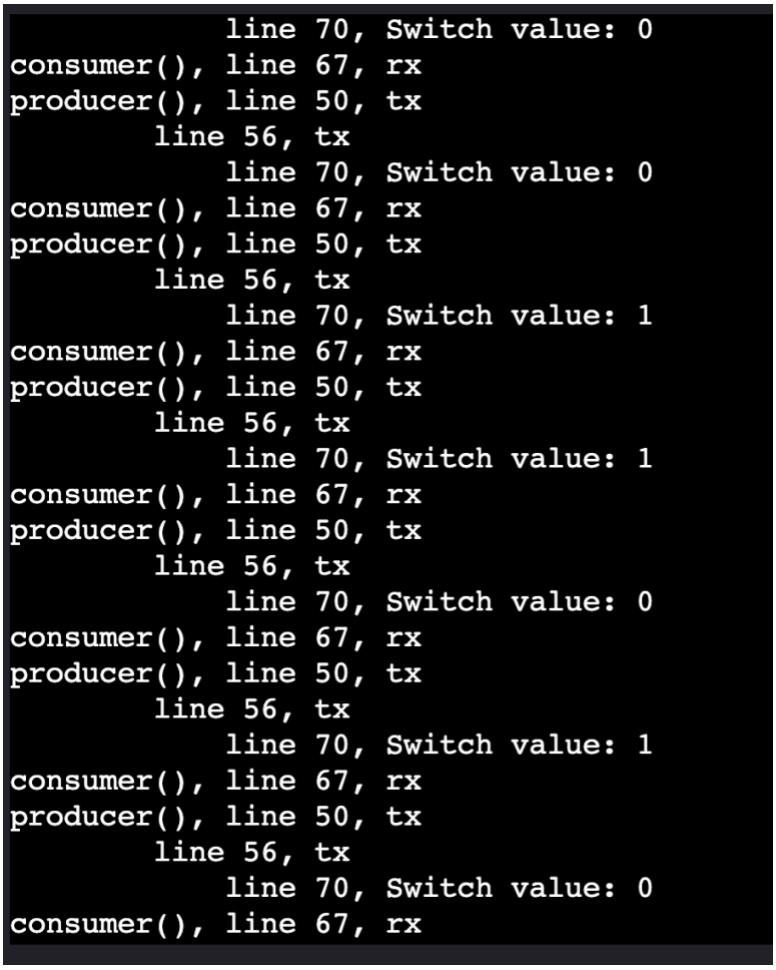
Producer: HIGH PRIORITY task
Consumer: LOW PRIORITY task

```
producer(), line 50, tx
        line 56, tx
            line 70, Switch value: 0
consumer(), line 67, rx
producer(), line 50, tx
        line 56, tx
            line 70, Switch value: 1
consumer(), line 67, rx
producer(), line 50, tx
        line 56, tx
            line 70, Switch value: 1
consumer(), line 67, rx
producer(), line 50, tx
        line 56, tx
            line 70, Switch value: 0
consumer(), line 67, rx
producer(), line 50, tx
        line 56, tx
            line 70, Switch value: 0
consumer(), line 67, rx
producer(), line 50, tx
        line 56, tx
            line 70, Switch value: 0
consumer(), line 67, rx
producer(), line 50, tx
        line 56, tx
```

Explanation:

1. As Producer task is of higher priority, it starts is execution and prints line 50
2. The producer task sends the queue with the switch value. But here xQueueSend() doesnot perform context switch as Consumer task performing xQueueReceive() is of low priority.
3. On successful xQueueSend() the Producer task, prints line 56.
4. When Producer task goes into vTaskDelay of 300ms, Consumer task takes the CPU and perform xQueueReceive().
5. On successful xQueueReceive(), the Consumer task prints line 70 with the received switch value.
6. As xQueueReceive() sleeps for portMAX_DELAY and Producer task's delay is not complete, the Consumer task prints line 67.
7. Later the context switch happens as high priority Producer task becomes ready.

Producer: HIGH PRIORITY task
Consumer: HIGH PRIORITY task



Explanation:

Here as both the tasks let's consider 2 scenarios:
Scenario1: Scheduler starts Consumer task before Producer task

1. Consumer task prints line 67 and prints Queue receive fail initially.
2. Later when Producer task becomes ready, it prints line 50 and sends the queue with switch value
3. Upon successful xQueueSend(), the producer task prints line 56.
4. When Producer task goes into vTask delay of 300ms, the Consumer task starts executing.
5. As the queue is not empty, the Consumer task performs xQueueReceive() and prints line 70 and the switch value.
6. As the xQueueReceive() sleeps for portMAX_DELAY, the Consumer task continues to print line 67.
7. Further when Producer task becomes ready the context switch happens.

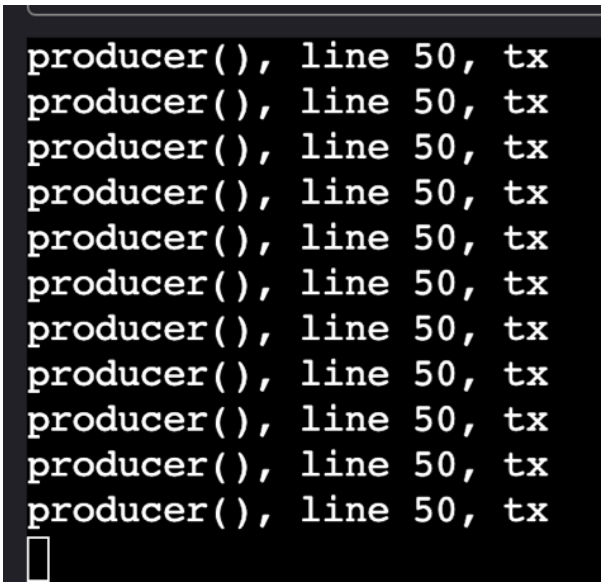
Scenario2: Scheduler starts Producer task before Consumer task

1. The Producer task prints line 50 and sends the queue with switch value
2. Upon successful xQueueSend(), the producer task prints line 56.
3. When Producer task goes into vTask delay of 300ms, the Consumer task starts executing.
4. As the queue is not empty, the Consumer task performs xQueueReceive() and prints line 70 and the switch value.
5. As the xQueueReceive() sleeps for portMAX_DELAY, the Consumer task continues to print line 67.
6. Further when Producer task becomes ready the context switch happens.

Additional questions:

1. What is the purpose of the block time during xQueueReceive()?

Answer:
The block time in xQueueReceive signifies that CPU is willing to wait for that particular duration to receive queue item, else xQueueReceive is failed.
If the queue has item the block time will not come into picture, but of the queue is empty the CPU will wait for the maximum of block time to receive an item. But if the item Is not received in that duration the xQueueReceive will fail.
Here, if the xQueueSend is commented, the item will not be received by the consumer task where xQueueReceive has block time of portMAX_DELAY so the Consumer task will keep on running in while loop for infinite time or till any queue item is received.



2. What if you use ZERO block time during xQueueReceive()?

Answer:

When Zero block time is used for xQueueReceive(), that means the queue receive does not have any block time if queue item is not received.
Therefore the xQueueReceive() fails as item is not received in 0 max block time.

```
consumer(), line 69, rx
Queue receive failed
consumer(), line 69, rx
Queue receive failed
consumer(), line 69, rx
Queue receive failed
consumer(), line 69, rx
Queue receive failed
consumer(), line 69, rx
Queue receive failed
consumer(), line 69, rx
Queue receive failed
consumer(), line 69, rx
Queue receive failed
```