Task 2

```
//thread place_order(data)
{
while (true)
{
order = OrderFactory.produceOrder(data);
if (self.queue.length() == MAX_QUEUE_SIZE)
{
// queue is full. Do nothing until somebody wakes us up
sleep();
}
self.queue.insert(order);
if (self.queue.size > 0)
{
// we have at least 1 element to process, notify the other comp
wakeup(process_order);
}
}
}

//thread process_order()
{
while (true)
{
if (self.queue.length() == 0)
{
// queue is empty, block until there is at least 1 element
sleep();
}
// if the queue was full, and we just made room, notify the pro
if (self.queue.length() == MAX_QUEUE_SIZE - 1)
{
wakeup(place_order);
}
order = self.queue.pop();
do_actual_processing(order);
}
}
```

❓ a. Assume that there are automated tests that take a string describing the operation sequence of the two components: "place_order(data1), place_order(data2), process_order(), process_order(),.. " (or in its abbreviated version: "P, P, C, C, ...")

Describe the most relevant situations (test cases) to be tested. For these test cases, mention the input that generates them (assuming MAX_QUEUE_SIZE=4)

P (place_order)

C (process_order)

Threads typical TCs

Aa TC	Input	Expected Result
1. Sanity test: place and process 4 orders	“P, P,P,P,C,C,C,C”	* orders placed into a queue * orders processed without issues and queue blocking
2. Queue with max capacity	“P, P,P,P,P”	* P is sleeping and stopped placing new orders as the queue is full
3. Queue =0	“C”	* C is sleeping
4. Checking sleeping /waking up and vise versa	“P, P,P,C,C,P,C,P,P,C,C”	* check that threads waking up/sleeping without deadlocks

💎 b. Analyze the previous pseudocode, and identify errors (if any). In case no error is detected, but if you would like to suggest improvements, mention them.

Potentially these `if` and `insert` operations can be interrupted by other threads. As an example, during insert operation `self.queue.length` can be changed to max size by another thread and this could result in inconsistent queue states, with more elements than allowed, leading to unpredictable behavior (a bug).

```
if (self.queue.length() == MAX_QUEUE_SIZE)
{
// queue is full. Do nothing until somebody wakes us up
sleep();
}
self.queue.insert(order);
```

the same for

```
if (self.queue.length() == MAX_QUEUE_SIZE - 1)
{
wakeup(place_order);
}
order = self.queue.pop();
```

---

c. If there are N and N instead of 1 order generator and 1 order processing, sharing the same
queue, what new situations that prior tests do not cover should be considered? Detail how you
would implement tests and describe a typical error.

In a system with multiple producers and consumers sharing a queue some orders can be missed:

- Producer 1 (P1) and Producer 2 (P2) both try to add orders to the queue at the same time.
- **P1 and P2 check the queue** and see that there's space available.
- **P1 and P2 both attempt to insert orders at the same time**, but because they are both using the same shared resource one of the orders might get lost or overwritten.

Same for removing orders (C1, C1,Cn)