Exercise 1: Warehouse System (Queue and Message Passing)

Scenario

A smart warehouse automation system manages inventory picking, packing, and shipping operations. The system consists of

- **Inventory Picker**
  a. Picks items from storage based on orders received and places them on a conveyor for packing.
  b. Simulates order reception via a queue (orders from external systems).

- **Packing Station**
  a. Receives picked items and packs them for shipping.
  b. Tracks how many items have been packed.

- **Shipping Controller**
  a. Reads packed orders and directs them to the appropriate delivery route.
  b. Uses message passing (via queues) to interact with other components for updating the status.

- **Logger System**
  a. Logs the status of orders, including when they are picked, packed, and shipped.

Requirements

- **Order Queue**
  - Use a FreeRTOS queue to simulate orders received by the system.
  - Ensure proper communication between the picker, packer, and shipping tasks.

- **Message Passing**
  - Use message passing via queues to simulate handovers between tasks (e.g., packing station sending completed orders to shipping).

- **Synchronization**
  - Use synchronization mechanisms like semaphores to handle critical sections or ensure tasks run in sequence when required.

- **Fault Handling**
  - Detect and handle scenarios like the order queue becoming full or empty.

Hints for Implementation

1. Use a queue to simulate a stream of incoming orders. Each order can be a structure containing an order ID, item details, and priority.
2. Use another queue for passing messages between packing and shipping controllers. Each message could include the packed order ID and status.
3. Ensure the picker waits for the queue to have orders before attempting to pick.
4. Use semaphores to control access to shared resources or coordinate between tasks.
5. Implement checks for queue full/empty conditions using uxQueueSpacesAvailable() or similar APIs.
6. Introduce mechanisms to retry operations or log faults for monitoring.
7. Use a dedicated low-priority task to log system status periodically.