CS5103: Assignment 11 // Date: 30-10-24

Question 1: **Warehouse and Delivery System**

You are required to create a warehouse and delivery system where multiple suppliers deliver items to a shared warehouse, and multiple delivery agents pick up items to fulfill orders. The warehouse has a limited capacity, with a fixed default of 5 slots for storage. If the warehouse is full, suppliers must wait; if it is empty, delivery agents must wait. The warehouse size should be taken as default (5 slots), while the number of suppliers and delivery agents is user-defined (each fewer than 10). Each supplier adds random items to the warehouse, and each delivery agent retrieves items. The simulation should handle a total of 20 transactions, using the `pthread` library for thread creation, semaphores to manage access to storage slots, and mutexes to prevent race conditions.

Hint: Use two semaphores: one for available slots (initialized to the warehouse size) and one for stored items (initialized to 0). Additionally, use `pthread_mutex_t` to lock the storage during additions and removals.

Sample Input:
Enter number of suppliers (<10): 3
Enter number of delivery agents (<10): 3

Sample Output:
Scenario: 3 Suppliers, 3 Delivery Agents, 5 storage slots, 20 total transactions.

Supplier 1 delivered item 42 to warehouse slot 0
Current Warehouse State: [42, -, -, -, -]
Supplier 2 delivered item 57 to warehouse slot 1
Current Warehouse State: [42, 57, -, -, -]
Delivery Agent 1 picked up item 42 from warehouse slot 0
Current Warehouse State: [-, 57, -, -, -]
…and so on until all transactions are complete.

Question 2: **File Logger Simulation**

Implement a file logger simulation where two processes, Process A and Process B, write log messages sequentially to a shared log buffer. Each process writes a fixed number of messages (take user input), and each log entry includes the process ID and a message number (e.g., “Process A: Message 1 - Task completed”). Only one process can write to the buffer at a time; use a mutex to ensure exclusive access, preventing race conditions. Use `pthread` to create

threads for Process A and Process B, and use a mutex to ensure messages are logged sequentially without overlap.

Test Input:
Enter number of Messages per Process: 10

Sample Output:
Process A: Message 1 - Task completed
Process B: Message 1 - Task completed
Process A: Message 2 - Task completed
Process B: Message 2 - Task completed
...
Process A: Message 10 - Task completed
Process B: Message 10 - Task completed