Lab7\task\answers.txt

1. There is no single “server” in a publish-subscribe system to store all tasks. Where should tasks be stored in a pub-sub system such as this?

in MQTT there is only clients (subscribers and publishers) and the broker.
the broker only forwards the massages to relevent subscribers, therefore tasks should be stored at each subscriber that needs them.

2. Who should generate task IDs? How should they be generated to avoid collisions?

The client that creates the task (publisher) should generate the task ID.
to make them unique, we can either use "timestamp + client_id" as the task ID(this approch still can make same id twice(very unlikely))
or we can use python built in UUID(uuid4) to generate unique IDs.

3. How can we represent the Task API ADD and DELETE operations in a pub-sub system like MQTT? Under what assumptions would the LIST operation be required?

to ADD;
publish: { "id": "1234", "state": "open", "description": "New task" } like massage to,
topic: tasks/add

to DELETE
publish: { "id": "1234" } like massage to,
topic: tasks/delete/{task_id}

4. When implementing the Task API operations on MQTT, comment on if, and how you would use the following MQTT features.

a. Quality of service (0, 1, 2).
QoS 0 (At most once) : Not reliable, not recommended.
QoS 1 (At least once) : Good for ADD, DELETE, and EDIT, ensures delivery but may cause duplicates.
QoS 2 (Exactly once) : Best for EDIT (state updates), prevents duplication but adds overhead

b. Clean session flag on topic subscription.
False (Persistent session) : Best for subscribers that need to reconnect and retrieve missed messages (e.g., task managers).
True (Clean session)   : Suitable for temporary clients that don’t need past messages.

c. Retained flag on message publication.
Use for tasks/add   : Ensures new subscribers get the latest task without waiting.
Use for tasks/edit/{id}   : Keeps the latest state of a task available.
Don’t use for tasks/delete : Deleted tasks should not persist.

5. Two students are arguing about how to structure topics
Student a suggests OPERATION/ID/STATE
Student b suggests STATE/ID/OPERATION
Either choose one of these proposals or suggest your own scheme stating your reasons.

Student a's structure is the best here.
because: Logical order : Action (OPERATION) comes first.
            Easier filtering : Subscribe to add/# for new tasks.
            More flexible : STATE can be optional.

Not STATE/ID/OPERATION (Student b) : Harder to filter, less intuitive.

6. Student c suggests including description in the topic as well. Argue for or against this suggestion.

I am "against" adding description to the topics.
Topics should be short & structured, but descriptions can be long and unpredictable.
Inefficient filtering, Harder to subscribe to specific task updates.
Wasted bandwidth, MQTT topics are meant for routing, not carrying data.
Better alternative will be to include description in the massage payload insted:
    { "id": "1234", "state": "open", "description": "Fix login bug" }