SYSC4001_A3_P2 - Design Discussion - Critical Section Requirement

George Tzemenakis - 10296691
Github Handle: TheRealRisu

Nicky Fang - 101304731
Github Handle: NickyF30

Overview:
This system contains several critical sections where race conditions can occur because multiple TA processes operate on shared memory at the same time. These include modifying the rubric, marking questions, and loading exams. Semaphores are used to enforce correctness using the three classical requirements for solving the Critical Section problem: mutual exclusion, progress, and bounded waiting.

Mutual Exclusion:
Mutual exclusion requires that only one process access a shared resource at a time when writing.

This is implemented through semaphores in the shared memory structure:

```
    sem_t rubric_mutex;
    sem_t rubric_read_count_mutex;
    int rubric_read_count;

    sem_t question_mutex[MAX_QUESTIONS];
    sem_t exam_loading_mutex;
```

sem_t rubric_mutex;
- Ensures only 1 TA can modify the rubric at a time

sem_t rubric_read_count_mutex;
- Protects the reader counter when multiple TAs access the rubric concurrently.

Together, these implement a reader - writer model, where multiple TAs may read the rubric, but only one TA can write to it at a time.

sem_t question_mutex[MAX_QUESTIONS];
- Each question has its own semaphore, meaning no two TAs can mark the same question at the same time.

This prevents duplicate marking or overwriting.

sem_t exam_loading_mutex;
- Only one TA may load a new exam file into shared memory at a time.

This prevents race conditions during exam transitions.

Progress:
Progress means that if a shared resource is free, a waiting process must not be blocked.

This design ensures progress because:
- Semaphores are only held briefly.
- No busy waiting occurs.
- No shared resource is locked unnecessarily.

Once a semaphore is released, the next waiting TA may proceed.

```
    sem_t exam_loading_mutex;
```

With this semaphore, once the exam is loaded, waiting TAs may continue execution.

Bounded Waiting:

Bounded waiting ensures no TA is starved.

The design avoids starvation because:
- All TAs have equal priority.
- Critical sections are short.
- No TA holds multiple semaphores at once.

Per-question locks reduce contention and isolate delays to one question instead of the entire exam. Meaning TA's can act together on the same exam, but never on the same question.