# **Lab 6 DVA454**

Jacob Johansson

Mudar Ibrahim

Question 1:

The implementation without semaphores has two problems that we found:

* **Race Conditions**: Without synchronization, the producer and the consumer could access the shared buffer when the other is reading/writing from/to it, leading to data corruption and unexpected behaviour.
* **Data Inconsistency**: Without mutual exclusion, the consumer could sometimes access the data that was currently being written by the producer, leading to inconsistent data retrieval.

Question 2:

* Fixing Race Conditions: We implemented a mutex semaphore to ensure that only one thread, either the producer or consumer, could access the shared buffer at a time.
* Fixing Data Inconsistency: We implemented ”empty” and ”full” semaphores. The ”Empty” semaphore kept track of available spaces in the buffer, preventing consumer from reading when the buffer was empty. The ”Full” semaphore ensured that the producer didn’t write to the buffer when it was full.

Question 3:

For the assignment 3, we had three different queues for every analog sensor (joystick, accelerometer and microphone), where each sensor had their own task to handle the reading. To ensure that the values were updated and that the gatekeeper never did the moving average of the values in the queues, we set the capacity to exactly the same number of required values for averaging described in the task. The gatekeeper waited for the queues to be full, meanwhile the other tasks, we call them sensorTasks, wrote values to each queue respectively only if it was not full. When all the queues were full, the gatekeeper task used the values, emptied the queues and calulcated the average to then send via the UART to print on the terminal.

En bild som visar text, skärmbild

Automatiskt genererad beskrivning