# Reference Monitor Writeup

After making modifications to the initial reference monitor in the first week and subsequently creating attack cases for the same reference monitor in the following week, I identified some mistakes in my original implementation.

The first mistake was a lack of concurrency handling. I failed to implement locks to ensure that one operation was completed before another could take over. This issue became apparent when multiple threads were executed on the same reference monitor. To address this, I incorporated the createlock() method and used the acquire and release methods as needed to manage concurrent access.

The second mistake was not handling the "undo" operation when the file was closed. In my initial code, I assumed that internal layers of the system would handle the "undo" operation when a file was closed and didn't properly handle exceptions when they were raised. In my revised reference monitor, I introduced a specific clause to handle the "undo" operation gracefully when a FileClosedError exception was raised, essentially allowing it to do nothing in that scenario.

I also made efforts to adjust my reference monitor to match the exceptions that people were checking in their attack cases. This involved substituting custom exceptions with 'RepyArgumentError' and 'FileClosedError'. However, some individuals were not accepting these generic exceptions and were looking for custom exceptions. As a result, some of my test cases didn't pass due to the exceptions I had decided to use.

In summary, I did my best to enhance the efficiency and security of my reference monitor and to address a wide range of attack scenarios, but there were challenges in achieving consensus on exception handling.