**Reflection on SFT221 – workshop 6**

1. In this workshop, the combination of both log files and assertions helped uncover different types of bugs. Assertions were particularly effective in identifying logical errors and invalid assumptions about the state of the program. They immediately flagged any unexpected conditions that should not occur under normal execution, such as NULL pointers or invalid indices. For any of assert violation, the code immediately crashed and showed an error stating assertion failed which also included the module name violated assertion as well as the line number where it occurred. On the other hand, log files provided a comprehensive history of the program’s execution, making it easier to trace the flow and identify subtle issues like correct values being processed or unexpected behavior at runtime. I had included log prints for all the functions executed (how many times, with what value) that allowed me to verify that the program is indeed taking proper steps in performing it’s operation. Overall, assertions worked better for immediate errors as I could directly go to that line and fix whatever violation occurred while log files were more useful for understanding the sequence of operations and diagnosing complex issues.
2. I prefer using a debugger for real-time, interactive debugging because it allows stepping through the code, inspecting variables, and evaluating expressions on the fly. This interactive approach can be faster for pinpointing the exact location and cause of a bug. However, log files are invaluable in scenarios where real-time debugging is impractical, such as in production environments, distributed systems, or when dealing with non-reproducible bugs. In such cases, log files provide a persistent record of the program's behavior, which can be analyzed post-mortem. While debugging with log files might be slower than using a debugger due to the lack of interactivity, it can be equally effective if the logs are detailed and comprehensive.
3. Confidence in having found all the bugs was gained through a combination of comprehensive test coverage, the use of assertions, and thorough log file analysis. By writing extensive tests covering various edge cases and typical usage scenarios, I ensured that the code was exercised under different conditions. Assertions helped catch logical errors early and log files provided insights into the program's runtime behavior. Despite these measures, I would not bet my life savings on having found all the bugs, as software systems can be complex and unpredictable. To increase confidence, I would conduct code reviews, perform static code analysis, employ continuous integration with automated testing, and seek feedback from users to catch any remaining issues. Additionally, implementing monitoring and logging in production could help detect and address bugs that only manifest in a live environment. However, I have lots of assertions in place for just these kinds of scenarios, I am highly confident that these will be very effective in stopping the execution if violations do take place.