1. I use GDB (GNU Debugger) with the help of my IDE's built-in graphical console as my primary debugging tool. I consider it the most effective tool for debugging. I can place breakpoints to step through my codebase line by line. This support allows us to view the values of variables on our main thread, which is particularly useful during multiple iterations. In this situation, printing out values would be relatively harder to trace hidden bugs or understand the dynamic changes between each iteration compared to using a debugger, because we can see all variable values at any moment.
2. Although I applied multiple techniques while do debugging this time, I didn’t use log files to trace bugs. The reason is this codebase is small and only run on a single machine once a time. Moreover, using log files to trace bugs often requires a proper infrastructure setting beforehand or external dependencies. And that doesn’t seem reasonable for this small codebase. However, logging files could be useful when we are working on distributed programs. At that time, we cannot break out our execution into one step or printing out a sequence of messages because there are many programs running at the same time. Also, another good reason for logging files is that won’t interfere our production program, but that is not the case we are dealing with now.
3. A debugger is designed for debugging, so it provides multiple features to help us resolve bugs, such as viewing context, stepping over execution, and checking the call stack, etc. Some of these features are almost impossible to achieve with print statements. Additionally, I prefer not to use print statements in my codebase. The reason is that they can clutter our organized codebase, and there is a risk of leaving these statements in the codebase after it is delivered, which might cause security or performance issues.
4. To test the program and fin the bugs, I use GDB with a tool providing a graphical console. I place multiple breakpoints and step over my code line by line to closely inspect variable values and execution flow to trace and identify the bugs. Also, I analyze the program’s execution. I can hypothesize some boundary case where bugs might occur. Additionally, I created a few extra test strings with various patterns of whitespace, numbers, and line breaks to ensure the code handled all cases correctly. This process bring me confidence to ensure that code can work correctly and I have fixed bugs.