For this programming assignment, I had done the basic testing on the GitLab testing scripts provided by the professor and included additional unit tests on my local machine as well as on the Ubuntu virtual machine.

Question: How does the code for handling a file differ from that for handling standard input? What concept is this an example of?

Answer: The main difference between handling a file from handling input from standard input is that for a file, one will need to implement the open() function before they can read the file due to the fact that a file descriptor is required for reading.

As for handling standard input, the open() function can be entirely skipped as standard input by default have the file descriptor of 0, meaning that it is pointless to open() the stdin as it is by default assigned a file descriptor of 0.

Furthermore, the code will have to look out for the EOF character in standard input, but does not for a file due to the case that read() will exit the loop once it notices that there are no more bytes to be read. However, read() will continue reading stdin since hitting return would return a byte, and it does not exit out of the loop if CTRL+D is inputted. So the solution will be to look out for the case where the number of bytes returned from read() is 0, and then exit.

Both of these differences are an example of the Abstraction concept where components are broken at a logical point. For example, reading from a file requires two separate methods: open() and read(). This can be seen as a logical break point for the bigger component of opening and reading a file. Furthermore, the input which is the file for open() is treated tolerantly as one can throw in any file they wish to open. However, the return value which is the file descriptor is treated strictly as that is the value we absolutely care about. This can also be seen as a black box since we do not care what happens while the open() method is processing, but only care about the file descriptor returned from inputting the file.