The testing I did for this assignment was both testing individual parts as I went along, and testing the system as a whole in different scenarios. Starting out, I would see if I could get anything to print because I was rusty on c++. So I started out with a simple “hello, world” program. Then I actually tried figuring out how to handle the case when the user inputs ./dog - or ./dog. I figured out how to take in input and print it out and tested that for a while. Then when I got that working, is when I went to handling files, and I would test this feature individually until I got this part working. At this point, I had a rough outline of the full system, and that’s when I moved to testing the system as a whole. For whole system testing, I tested text files with one line, multiple lines, combination of dashes and files, and then binary files. The way I tested binary files was using the diff command and seeing the difference between cat’s output and my output. And finally I put in error handling and would make sure the program would still run if an error was thrown with a file.

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

The code differs because file handling goes until it reaches the end of the file and prints, while handling standard input goes on and on and on until the user gives a termination signal. File handling also requires error handling in case opening, reading, writing, or closing the file runs into a problem. Because you have to handle these two different cases, it naturally splits itself into two components which increases the complexity of the system. The way I split it up is abstraction because it’s only logical to separate handling files and handling user input into two different functions.