Testing:

I first started by implementing the part of the code to detect if the input is a file, a dash, or contained no arguments. The program only had dummy functions that contained print statements made for testing to see if the program was going into the right functions depending on the arguments.

After I completed the testing on the main, I wrote a function for file printing and tested that by calling the function on multiple different files and running diff(1) on ./dog [filename] > [targetFile] and cat [filename] > [targetFile]

After testing the file function, I wrote a function for reading from STDIN and tested it by running diff(1) on ./dog - > [targetFile] and cat - > [targetFile]

After testing the STDIN function, I went back to test scenarios where the program should return an error and edge cases. Examples of these tests are:
Files that don’t exist
Directories
Files in other directories. (ex: ./dog ../../randomFile)

Lastly, to finish testing I used the functional test script to test my program.

Question:

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

A: The code for handling a file is different from that of handling standard input because the code for handling a file is a function that opens a file and pulls the data from that file and prints it out to STDOUT. The code for handling standard input is a loop that continuously accepts a stream of information and prints it out to STDOUT. This is an example of the concept of modularity because each function is it’s own module that performs a separate function that work together to form the overall system.