Mia Carter
030038422

Design:
The design of this program includes the use of a pipe, a fork to create the child process, the pipe ID to find which process we are in, a buffer to temporarily store contents, and reading and writing to and from those processes. I began the program by parsing the arguments: argv[1] is the source file, and argv[2] is the destination file that the source contents will be copied into. If the user does not appropriately pass the 3 arguments, an error is shown to tell them all arguments are necessary for execution. If the user attempts to pass invalid files, an error is shown letting them know which file is invalid. Then, a pipe is created of size 2 to account for the read and write ends of the pipe. If the pipe is not successfully created, it returns -1, so I handled that error and displayed a message. I then created the child process using fork and assigned it to the pid (pipe identification) to keep track of which process is currently running. If the pid is a negative value, then the child was not successfully created, and an error message is displayed. If the pid is 0, then we know we are in the child process. If the pid is greater than 1, then we are in the parent process. When we are in the parent process, I used a while loop that terminates once all the contents have been read until the end of the file is reached. The parent then writes to the write end of the pipe. When we are in the child process, I used a while loop until all of the contents of the

Individual Contributions:
This project was fully completed by me!

Link to code compilation: https://youtu.be/YmJDHwYfsYI