Jordan Mai
1645861

Design Document

1. **Project Description**
    a. Our program called *shoulders* will take in non empty files to be read in and copy the given amount of byes and at most 100 bytes of content into stdout. If no files are given for input, then the program will be read from stdin and copy to stdout. The program ends once all files have been read.
2. **Program Logic**
    a. PSEUDOCODE
        i. Read in argv[1] to consider number of bytes
        ii. For int i=2 to EOF:
            1. Open the file at argv[i] in the command argument
            2. Read in at max argv[1] bytes from the file
            3. If argument is '-' or empty then read from stdin
            4. Write out the file to stdin
            5. Close file
3. **Data Structure**
    a. An array works by allocating space for some bytes of content to be put into. A buffer array is used to store bytes read from a file. This array of characters will then be used to print to stdout.
    b. A file contains amounts of bytes. Files will be used for where the program reads bytes from.
4. **Functions:**
    a. int main(int argv, char*argv[]):
        i. This function is the main function for the program. It will take in arguments from stdin and read the files or inputs one by one. The program expects inputs of an integer(which determines the number of bytes for the program, and optional files to be read from. The program will read in each argv file and copy the maximum of bytes specified to stdout. If no file argument is inputted, the program will read from stdin in which the user types in things. This program will use a for loop to iterate through every command argument and open and read from each file to stdout.
5. **Question: How does the code for handling a file differ from that for handling standard input? Describe how this complexity of handling two types of inputs can be solved by one of the four ways of managing complexity described in Section 1.3.**
    a. The code for handling a file will need to have an open command to open the file to read from. Then it needs to close the file at the end. The code for handling standard input will not need the open and close commands because it can just read from whatever the user types in the stdin command line. For a file, teh code will need to check for errors in opening and ensuring file works, while for standard input, no error check is needed.

b. This complexity of handing two types of inputs can be solved using the layering to minimize the interaction between two modules. Because handling files is different from handling standard input, we can use the concept of layering to build off of previous modules or code to improve complexity. Within the first handling of file input, we will simultaneously call for handling standard input. The code will both accept file input and stdin as part of the layering complexity.