For the first part of our larger program project, we were tasked with creating a program that would extract the contents of a 10-k report file from a large company. We took these reports from the U.S. Securities and Exchange Commission website and their built-in EDGAR search functionality, which holds the filings of certain companies registered, including annual 10-k files. I decided to look into the 10-k reports of two large media streaming companies: NetFlix and Paramount Global.

The first challenge was figuring out how to transfer these reports into a file that could be readable by my program. Since the reports on the EDGAR webpage utilize XBRL, I could not simply transfer the contents of the report over to a local directory. There were no options in the web viewer to download the contents of the report to a specific file format. As a result, I opted to copy the contents of the report into a text file that I could store into a directory. This way, I could better utilize a file format that I was familiar with, and since the most important part of this program was mainly reading the string contents of the file, I felt that it was a sufficient enough solution for the problem at hand.

Another problem I faced was handling how to count the parts of the file. In the reports from both NetFlix and Paramount Global, they separated part sections in a similar format. That is, they began with the words “PART” and then the part number in roman numerals, and possibly a brief title summary. While the first word could be easily handled with a case-sensitive statement that checked for it, the part number in roman numerals meant I could not just check for a number. As a result, I decided to make another method that would take in an integer and output the corresponding number in roman numerals. This allows the program to be a bit more versatile without having to bloat the readFile method with statements that check for individual roman numerals. With the method I implemented, it also means that strings in different places with

similar structure (e.g. “PART II” in line 35, and “PART II” in line 47), will only count for one of them.

Finally, the other problem was checking for duplicates within the output file and allowing the writeFile method to be called multiple times without overwriting the contents of the file. If I appended the information to the output file like in the latter, then it would cause duplicates if the same text file had been passed previously. I created a method that would take in a string and check if that corresponding string was in the output file. If not, then the writeFile method would append the corresponding information, or output an error message and not write anything if otherwise.