Daniel Domalik
CIS3190
Michael Wirth
03/23/2018

Assignment 3, COBOL Roman Numeral Converter

To start, I will run through a description of how the program operates. When you run the program, it will display a short introduction as well as some rules. The user is then prompted for his/her input. The user can either type a roman numeral to be converted, or type a special character followed by a file name to convert a whole file. If the user enters a single line, the conv function is used to scan through the string and add each roman numerals respective value to a sum called sum-val. If the user enters a file (>filename.ext), this process is simply looped. It functions the same as if the user would manually enter everything in the file. Throughout the process, a loop is put in place to ensure that all values in each inputted string are valid.

A simpler version can be displayed like this:

1. Display rules
2. Prompt user for input
3. Check if input is valid
4. Proceed with conversion
5. Display inputted value along with converted value
6. Loop (through file or return to step 2)

The re-engineering process was quite pleasurable. The only thing I struggled with starting out was the variable declaration. It seemed so tedious. COBOL ensures that you enter all variables that you need right off the bat instead of perhaps declaring them down the line. All go to statements were removed as well as some write statements. God help whoever had to write COBOL like the original code. Everything seems even more tedious than it already does. Things like “after advancing 1 line” would be so much simpler in other languages. I declared a bunch of new variables to help keep track of everything. Some of these variables include is-valid (for checking if the string entered is valid), int (which is the sum of the converted numeral), loop (for helping to simplify the process).

The things I liked about COBOL was string manipulation. It had a “Python-esque” feeling to it. The reason I picked this out of any more possible things was because of one of the assignment specifications. The spec stated that the user should be allowed to read in a file containing any *x* number of roman numerals. For my program I allowed the user to do this by typing the ‘>’ character, followed by the file name. COBOL made this really easy for me because I could just grab the file name by doing input-filename(2:) just like you would in Python. What that does is take everything after the first character of the string. During development, I kept getting an error telling me that “>numerals” was not a valid file name. I was pulling my hair out until I decided to try my solution. Lo and behold, it worked. Instead of the file reader taking “>numerals” as a parameter, it was taking “numerals”, perfect! Something else related to string manipulation was the lower-case and upper-case part of the spec. The user should be allowed to enter both upper-case and lower-case roman numerals. This was extremely easy to do with a

simple function built in COBOL. One other thing I liked about COBOL was the file I/O. More specifically the I because I did not work with the O. COBOL, being invented 50 or so years ago, gave me quite a scare when thinking about how to do things. Thankfully, it was not as hard as I thought it to be. It was quite simple actually, again, similar to Python. A simple loop would do the trick. Read until “end-of-file”, while scanning each line perform the conversion and that requirement was finished. Finally, I enjoyed COBOL because of how human-like it was. Personally, it seems like a a programmer would use the English language to try to write code. Things like DISPLAY and OPEN INPUT are very English/Human-like. This being compared to something like C where printing something to the screen takes up all the space on a line in your IDE/text editor.

When there are good things, bad things always seem to follow. What I really disliked about COBOL was the variables. At my understanding, you cannot declare variables half-way through a function. It is modular, yes, and you can declare new variables when calling a function, but not in the middle of it. From a programming perspective, I can understand how this is a good thing and a bad thing. Some new programmers coming from languages like Python or C where you can declare variable anywhere you’d like may have issues and frustrations with this.

My knowledge of programming didn’t really do anything for me in terms of understanding COBOL. Like I stated previously, it is very English-like. That is what made it enjoyable to learn. Nothing about this assignment made me “dig deep” into my programming experience and try to come up with a solution. After the re-engineering was finished, everything else was straight forward. I felt like the computer was my friend and it understood English. When I wanted it to do something I would type it out just like a sentence and it understands. Really fascinating!

One thing I wish I could have accomplished was printing the converted numeral on the same line. Like most languages, after the user inputs something from the keyboard it carries over the carriage return. I would have liked to have the user input their roman numerals and print out the converted value on the same line. To get over this I had to re-print the user input along with the converted value. Sad times.

All in all, this assignment was pretty straightforward. The time it took to complete this assignment was roughly ~3 hours including the write up. The most aggravating part of it was the re-engineering process. After reading a few resources, I felt like I understood it. Considering COBOL is still widely used, I may take more time in learning it and seeing what it’s full potential is. Another language under the belt is always good. Who knows, maybe it will land me a job in the future!