CIS * 3190 – Assignment 3 Cobol Reflection Report

My Re-Engineering Process

1. Familiarizing myself with Cobol syntax

The first step to my re-engineering approach was to attempt to learn about Cobol and what the syntax means. In order to do this effectively, I went through the three packages that talk about Cobol on courselink. From these packages, I learned the basic syntax and structure that makes up Cobol programs and I was able to learn about the differences between Cobol 74 and 85.

2. Analyze existing code

After feeling comfortable with what I read, I proceeded to download the code provided to us from the professor. This code included the programs, romanA3_1.cob and conv.cob. In order to get an understanding of how this code worked, I went through each program logically and traced the steps of the program. This allowed me to gain a better understanding of how a roman numeral value calculator works.

3. Modernizing existing code

The next step was to start modernizing the code. I started by going through the existing code and changing it from uppercase to lowercase, and then by finding the certain syntax I could change to Cobol 85 (Ex. Picture -> pic, removing “conventional”, etc). One of the biggest changes I made in my opinion was to use “Display” instead of “write” and “Accept” instead of “read”. This simplified input and output for me and allowed me to accomplish a lot of the tasks that were very complicated in the original code, in a few, easy to understand lines. A few other examples of the changes I made were that I got rid of the “go-to” statements and created many subroutines (paragraphs) to improve modularity and readability.

4. Re-writing existing code

In order to develop code to my liking, I decided to re-write the code from scratch, using the same flow/logic that was used in the files we were given. I began doing this by commenting out the code below and then writing new code in Cobol 85 that accomplished the same tasks. By doing it this way, I was able to find what I thought was the best way to modularize the program and split it up into small paragraphs. Throughout the re-writing process, I used similar variables to the original and I decided to make the output I provide to the user look very similar.

5. **Additions**

As I'm sure you know, not all the functionality we needed was included in the cob files that we were given. Therefore, my next step was to add functionality, such as accepting and parsing files, accepting user input on one line and validating user and file input.

6. **Cleaning up, testing and commenting**

Towards the end of this process I went through the programs and removed all the unnecessary older code. I also added more comments where I felt there were necessary, including subroutine comments. Finally, I ran my code on the SOCS server against the test file given by the professor and one I built myself.

Was Cobol easy to learn?

In my opinion, Cobol was not a particularly easy language to learn in comparison to the ones I know. I say this because it is much different than any of the other languages I've used up to this point in my life so it was hard to pick up on the syntax and the structure. One of the many things that made Cobol hard to learn was the declaration of variables, as well as the division and section structure.

What structures made Cobol challenging to program in?

I found that Cobol doesn't offer much string manipulation options. I originally planned to use strings for getting the user input but I found it too difficult to loop through the string and get the index. I'm not sure if there is a way to do this but I researched for a significant amount of time and wasn't able to come to a solution. Furthermore, a second structure that made it challenging in my opinion was loops such as if and evaluate. It was difficult knowing when to use periods and when not to use periods in these.

What did you (i) like, (ii) dislike about Cobol?

I liked that Cobol 85 was much easier to use and read than its predecessors. It uses display which is essentially a simple print statement and I found that the syntax was much easier to read after the small changes like making it less wordy, changing picture to pic, or removing conventional after variable declarations. One of the things I disliked about Cobol is how all the variables had to be declared in the working storage section.

Synopsis of Cobol

Overall, I didn't like Cobol very much. Although modernizing it made it much easier to read and understand, there were still lots of things missing that other programming languages did very easily. This is why I would definitely choose it last if I had a choice between it, Fortran and Ada to accomplish a basic task.