CIS\*3190

Assignment 3: Cobol

Kyle Hersey

0851973

Steps to re-engineer:

- copied conv and roman code into new files

- spaced out the code so it is more legible

- start with conv because it has a simpler structure than roman

- replace loop through end-loop with a perform block

- replace b1 – b6 with else if statements, b7 as the else

- put code from 3 outside of the if - else if - else block

- add code for lowercase letters (example: if s(i) equals 'I' or 'i')

- replace all the b7 print stuff with a display statement

- should be ready to test

- make a new cobol file

- write a loop to get input, call conv and print the converted value if the return code is 1

- compile and test

- if – else if – else in conv is giving me all kinds of errors and wont compile

- replace if – else if – else block with an evaluate block

- compile and test, this time it worked

- move to roman

- replace all the WRITE STDOUT-RECORD statements at the start of the program with display statements

- replace the general GO TO L1 structure with a perform block

- replace all statements related to input with a display prompt and an accept statement

- we no longer need N (variable to keep track of number of characters entered) so remove it

- replace GO TO DEPENDING statement with an if statement, where if ret is 1 print converted value

- compile and test

- go to google and try to figure out how to use file I/O

- try to find something about using dynamic files (not hardcoding the file name)

- after about 30 tries finally find something

- implement prompt “would you like to use file input?”

- implement if – else for file input or terminal input

- implement code to get filename and read entries line by line, passing entries to conv for conversion

- compile and test

- clean up working storage, file division, and environment division stuff

- clean up working storage for conv

- done

Would this program have been easier to rewrite in scratch in a language such as C?

Yes, absolutely. Cobol is a really archaic language that feels supremely old, it is so different from any modern language, that as someone who knows a bunch of modern languages, I could have more easily and more quickly remade this is any of them.

What was the greatest challenge you faced in the re-engineering process?

For this program the hardest problem I ran into was figuring out how to use file I/O. We didn't learn how to use dynamic files and we need them for this project. While there is a plethora of Cobol resources, most of them are related to working with IBM Mainframe. A lot of terms I am familiar with in other language have older archaic names in Cobol and thus googling the answer to this one relatively simple problem took a total of about 2 hours. Other than that the rest of the re-engineering was relatively straight forward.

Is your program shorter or longer?

My program is quite a bit longer. I had to add all the file I/O, and I ended up just adding extra print statements to the code and spacing it out more to make it more legible.

What was your overall opinion of Cobol?

Even after learning Fortran and Ada I cant help but thinking that Cobol feels by far the oldest of the three. I am really not a fan of the structure it imposes on programs. Cobol seems to function in a fundamentally different way under the surface than most other programming languages I have used, and that really threw me off. The way variables are defined has the upside of being very memory efficient but has the downside of being really ugly and quite cumbersome to work with. I've never disliked a language before, but I hate Cobol. Hopefully I will never have to program anything in it ever again. I would advise anyone else who hasn't used it to avoid Cobol like the plague.