Reflection Report

Cobol was a difficult language to learn as its syntax is unlike the previous languages I am familiar with such as Java, python, and C. However, Cobol is a self-documenting program which allowed me to easily adapt to the language and efficiently complete course activities and this assignment.

The file input and output structure made Cobol challenging to program as it was unique from any other I/O structure I have used. Firstly, Cobol has the option to set the organization of how a file is read, and in the case of this assignment I had set this to “line sequential” allowing me to read each line in a file individually. Secondly, Cobol uses what are known as records which is a collection of variables that store data. Since this concept was completely new to me, I found this to be a struggle when programming and accessing data. Although this was difficult to learn at first, it made my program more efficient and error proof in the end. Moreover, I had difficulty when attempting to check if a file exists in the current working directory when prompting the user for the name of an ASCII file. This required me to do extra research into academic sources to fully understand this procedure. Cobol has a keyword named “call” where I am able to call and check the value of a return code, where I can base my conditional statements on the value of the return code. Lastly, a difficult usage feature of Cobol is its “subprograms”. According to the reading on CourseLink labeled “Intermediary Cobol”, a procedure division paragraph is a subprogram and subprograms can be both internal and external. This terminology along with the syntax of this feature caused me countless errors and warnings from Cobol’s compiler when first attempting to create subprograms. Once again, I was able to learn this concept through extensive research of secondary resources and eventually understanding the generic structure of Cobol’s use of subprograms.

In contrast to the above issues, I found multiple features of Cobol useful that other programming languages generally don’t have. To begin, I enjoyed how Cobol is self-documenting as I found it useful for understanding expressions for conditions and loops. When working through the algorithm of this assignment, I wrote a basic pseudocode program to solidify my understanding of the ISBN validity program. Multiple lines I wrote in pseudocode such as “if EOF is not zero” and “perform valid-check until j is less than 10” were almost written identically in Cobol. This made the process of writing the program quicker and more efficient. Additionally, I found going back to my program to continue working on it was made efficient as I was able to read and understand the code through its self-documentation. I also enjoyed Cobol’s unique method of declaring variables. I enjoy how you can set the exact number of digits as well as set where an assumed decimal point may be. This made the assignment easier especially when requiring to read the first 10 digits per line in a file and ignoring the rest. Furthermore, I enjoyed how all variables were global as they could be accessed through all subprograms.

Although I mentioned many structures that made Cobol challenging, following my research I was able to increase my knowledge of Cobol and it did not pose as much of a challenge as it had originally. The one feature I disliked, is the method used for accessing specific characters in an array. This gave me multiple issues as it was unlike any other language I have worked with previously. Although I completed this, it would not be my first choice for accessing individual indices in an array as I believe it is overcomplicated.

Ultimately, Cobol is a unique language that poses many new unique features and functionalities. Through the course readings and activities provided on CourseLink, I was able to solidify an advanced

Reflection Report

understanding of Cobol in order to complete this assignment and will consider using it for the final assignment in this course.