This reflection report will provide a synopsis on my experience with COBOL for my third assignment of this course. Before I begun implementing my ISBN validation program, I read through the A3 description and introductory and intermediate COBOL documents. I annotated anything I found interesting and made note of any structures I thought would be useful to incorporate in my code.

Once I familiarized myself a bit with COBOL, I created my isbn.cob file and put together the simplest program possible to test compilation. I then implemented readISBN to get an idea of how file reading worked. To my surprise, file reading was the easiest part of the assignment, though many other languages fail to impress the average programmer in that aspect. Instead of putting error handling for the input file off until I had finished the main functionality of the code, I took a different approach for this assignment and performed more thorough error handling as I added new functionality to the program.

I then created stub isValid and checkSUM paragraphs in which I wrote descriptive comments to brainstorm how I wanted to tackle adding the logic for the two paragraphs. I started off with what I found easiest first, which was adding error messages for alphabetic characters that were present in places they were not supposed to be in. I let "correct and valid" be the default option for the time-being for any unhandled case. I also added the "trailing/leading 0" and "trailing uppercase/lowercase x" message logic so all I'd have to work on later would be the "invalid check digit" message once checkSUM functionality was implemented. I added temporary *display* statements along the way to ensure variable values were what I expected them to be.

After adding all required functionality, I modularized my code and reduced comments for self-explanatory bits. I added more whitespace between paragraphs to make it more apparent that those blocks of code were separate and weren't necessarily going to be executed one after another.

To reflect on my experience with COBOL, I did not find it easy to learn. Of all assignments I've completed thus far, I had the toughest time with this language due to the unfamiliarity of the syntax, structure, and coding conventions.

One structure I felt made COBOL more challenging to program in was the array. For example, after a lot of time spent wondering why I was unable to access specific chars within my ISBNs, I learned a variable declaration like varName pic x(10) wouldn't let me access every char individually, but a declaration like varName pic x occurs 10 times would. I also had a tough time wrapping my head around 2d array declarations, but once I begun using them more, they started making more sense to me. Many other parts of COBOL failed to grow on me though. First, I don't enjoy being required to use *verbs* for calculations instead of symbols. Even when symbols are used, the *compute* keyword is required. I believe such keywords only make the code look bulkier. As a C and JS programmer, I also didn't enjoy using periods at the end of lines. In fact, these periods ended up being the most frustrating part of the assignment for me. In my *file-control* section of the *environment division*, I added a period on both lines 12 and 13 of my code, not realizing I only needed it on line 13 (refer to image 1 on page 2). The errors listed upon compiling my code mentioned everything but the extra period which was the actual issue. To witness the abominable consequences of misplacing a period in COBOL, please see image 2 below.

Despite the many challenges I faced with COBOL, some aspects of it caught my attention and I'd like to use the final part of my reflection to acknowledge them. First, I appreciate that COBOL is written in a way that can be understood by anyone reading the code; it's self-documented, so comments become less necessary to include for simpler programs. Second, I like that as programmers, we still have somewhat of an option to use symbols instead of keywords if preferred, such as > and < instead of *greater than* and *less than*. We also don't need to re-state variable names before relational operators if the condition applies to the same variable used previously in the same *if* statement. For instance: p > 0 && p < 8 could be condensed to p > 0 and < 8; this is something I found very convenient and wished more programming languages supported. For these reasons, while I still don't prefer coding in COBOL, I can understand why it was widely used in the past.

Image 1: root of problem

```
12  select input-file assign to dynamic ws-fname.
13      organization is line sequential.
```

Image 2: misleading compiler errors

```
rqureshi@percy:~/Desktop/CIS3190/A3/CIS3190_A3$ cobc -free -x -Wall isbn.cob
isbn.cob: 13: warning: 'ws-fname' will be implicitly defined
isbn.cob: 13: error: PROCEDURE DIVISION header missing
isbn.cob: 13: error: syntax error, unexpected ORGANIZATION
isbn.cob: 15: error: syntax error, unexpected DATA
isbn.cob: 16: error: syntax error, unexpected FILE
isbn.cob: 17: error: syntax error, unexpected FD
isbn.cob: 18: error: unknown statement '01'
isbn.cob: 19: error: unknown statement '02'
isbn.cob: 21: error: syntax error, unexpected WORKING-STORAGE
isbn.cob: 22: error: unknown statement '01'
isbn.cob: 23: error: unknown statement '01'
isbn.cob: 24: error: unknown statement '01'
isbn.cob: 25: error: unknown statement '01'
isbn.cob: 26: error: unknown statement '02'
isbn.cob: 28: error: syntax error, unexpected PROCEDURE
isbn.cob: in paragraph 'readISBN':
isbn.cob: 39: error: 'feof' is not defined
isbn.cob: in paragraph 'read-file':
isbn.cob: 43: error: 'i' is not defined
isbn.cob: 44: error: 'feof' is not defined
isbn.cob: 46: error: 'isbn-record' is not defined
isbn.cob: 46: error: 'i' is not defined
isbn.cob: 46: error: 'isbns' is not defined
isbn.cob: 47: error: 'i' is not defined
isbn.cob: 48: error: syntax error, unexpected END-READ
```