This reflection report will provide a synopsis on my experience with Ada for my second assignment of this course. Before I begun implementing my image processing program, I read through the entire assignment document to better understand the program requirements. I then read through the introductory, intermediate, and advanced-level Ada resources to familiarize myself with Ada syntax and get an idea of which structures I could use for my program.

My first step toward a complete project was to create and connect all packages with the main wrapper program *image.adb*. To confirm whether the wrapper program was calling all required subprograms correctly, I added put_line statements in each subprogram and ensured all print statements were displayed once I ran the program. Once all packages were connected, I implemented the complete functionality for *getFilename()*, storing the input and output filenames and performing error handling for invalid inputs. I then stored the image file contents in temporary variables to later save in a record once image contents were validated. Upon completion of my *readPGM* functionality, I looked toward creating a more interactive UI by adding a menu, allowing the user to choose which transformation they wanted to apply to their image once they'd entered a valid input file. I then completed my *writePGM* subprogram and tested my code to ensure all subprograms were working as expected.

After all subprograms were implemented, I worked on reducing the number of temporary variables used in my program. I modified my code to store values read directly into my image record. I also set my modified image record equal to the original image record before passing it into image transformation subprograms, so the original image record wouldn't also have to be passed into each of them; the transformation could be applied directly to the modified image record I'd created.

While the following may be subjective, I found string operations, specifically dividing strings into substrings, to be unnecessarily difficult. This was quite a significant drawback for me, since storing contents of the input image file required being able to break lines of code into smaller substrings, and then converting them to integer values. Other programming languages, such as Java, merely require a split() method to implement the same functionality, and with less lines of code. A relatively less significant drawback in the early stages of implementation was that Ada indexes from 1, unlike C, which initially led me to declare my histogram-related arrays with invalid bounds (1..256 instead of 1..257). I caught my mistake faster than I normally would have, though, as Ada has more comprehensive error messages, and I learned from my experience with indices in Fortran for Assignment 1. I also had a tough time with I/O operations, as I found Ada to be stricter about how variables that are not strings must be formatted to be displayed using put and put_line. I came to appreciate this aspect of Ada though, as it helped me ensure only values I intended to be printed were displayed to the user.

Though wrapping my head around some Ada structures was quite difficult initially, I began understanding its benefits once I started implementing my program. For example, Ada if statements use = for comparison, which goes directly against what C does (= = for comparison and = for assignment). Ada's syntax made it harder to mistake an assignment for a comparison check and vice versa, since assignments used := instead of merely the = sign, which is also the only symbol used in C's equality operator. Second, Ada has structures that encourage code modularity, such as packages, which provide a mechanism for organizing complex or large programs. Ada also prioritizes readability over conciseness; keywords are not abbreviations, so oftentimes it is easy to understand what code snippets do without conducting additional research. It is also easy to introduce new types in Ada. Since there is barely any type inference, programmers must explicitly declare types being used, which helps prevent data usage errors. For these reasons, I believe Ada was well-suited to solve the image processing problem presented to us in this assignment.