CIS * 3190 – Assignment 2 Ada Reflection Report

My Approach

1. Building the dictionary

During the planning of my algorithm, I decided that logically, I would make the first step of my program to load in the dictionary from the built in "canadian-english-small" in the "/usr/share/dict/" directory of my computer. The reason I did this is because then I only wanted to load it in once and I was able to reference the dictionary at any point in the program going forward. For the data structure that holds the dictionary I decided to use an ordered set of unbounded strings. I was able to do this by declaring the package for ordered sets with unbounded strings as the type of variable it contains in the main "solveJumble" procedure. The advantage for using this type of data structure was that I was able to use a contains function to search through it when necessary. As far as how I loaded in the dictionary, I looped through the file line by line and inserted one word at a time into the ordered set.

2. Getting user input

The next step was to get the users input. For this task I decided to use a procedure as I wanted both an array of the input and the number of words to be returned to the main procedure. For the input, I stored it in an array of unbounded strings that was given a max input size of 200. I decided on this number because the declaration of the array is static in my program and I needed a size that would be reasonable enough that no one would enter that many word jumbles. As far as handling user input, I did not remove any white space in the jumbles they are entering as I thought it was unnecessary given that word jumbles should be words and not letters spaced out. Lastly, I decided to end the user input when the user clicks enter on a blank string. This gives them the ability to enter multiple words on different lines.

3. Getting all unique anagrams of the users input

After getting user input I had to find all the anagrams for the words they entered. To do this I first went through each word at a time and got the factorial of the length of the word. This allowed me to dynamically size an array of strings to the number of possible anagrams that word has. Then I passed the users input into a generate anagrams function which permuted the word recursively and returned all the possible anagrams that were unique in an array of unbounded strings. I decided to remove duplicates of anagrams because it made sense to me and it removed duplicate answers in the final output.

4. Comparing anagrams to the dictionary

The final step in my algorithm was to compare all the anagrams that were created and stored in the array in step 3 against all the words in the ordered set that represented the

dictionary that I created in step 1. For this I did a simple loop through the all the anagrams and checked if the dictionary contains that word using a contains() function. If it contained it, I printed it to the screen beside the word jumble the user entered. I then finished by printing out the number of anagrams that were found in the dictionary.

Was Ada well suited to solving the problem?

I believe that Ada is well suited to solving the problem because it included all the necessary data structures and types that we needed to complete this assignment. The built in unbounded_string/string type was easy to use and allowed for easy string manipulation, unlike programs without predetermined string types like C for example.

What particular structures made Ada a good choice?

The data structures that made Ada a good choice included arrays of strings (both regular and unbounded) as well as the ordered set. The ordered set made it easy for me to store the dictionary as all I had to do was insert it and it would add the back of the set. It also made it easy for me to search the set after it was created because it provided a contains() function. I used an array of unbounded strings to store the anagrams and user input instead of an ordered set because I found it easier to iterate through an array and I had no need to search either of those arrays by specific strings.

Benefits / limitations?

Some benefits to Ada include:

- A variety of useful and common data types
- Built in data structures such as arrays and ordered sets
- Easy to use functions and procedures (Consistent with Fortran)

Some limitations/negatives include:

- No block comments
- Only can declare variables at the top of the procedure/function or in a declare block
- Inconsistent syntax to other languages (Ex. ":=" instead of "=" or the type after the name in declarations)

Synopsis of Ada

Overall, I had mixed feelings about Ada. On one hand I found it easy to come to a quick and easy solution using the built-in data structures and types and I did so in minimal lines (140 without comments). On the other I considered it to be hard to read because of how all the

functions and procedures are intertwined with each other. Besides that, there were a few things I found change regarding syntax that I mentioned above. In conclusion, Ada is a programming language that I would probably not use over Fortran or C but it is not a bad language to program the right task in.