I decided in this program to primarily use unbounded string arrays instead of string arrays since I wouldn't need to worry about initializing arrays before I used them as the size of all these unbounded strings were all dynamically allocated instead of a defined size.

A basic explanation of my programming process was to essentially gain a basic grasp of Ada before starting the project, and then proceeding to focusing on getting the user input. I then proceeded to waste a lot of time trying to understand the recursive find anagram function before hastily putting together the finding word portion and submitting this assignment.

The approach that I took was that of to get a basic understanding of Ada before I proceeded to start working on the assignment. Afterwards I decided to proceed first with understanding how to get the user input and inserting all that information into an array. I continued to figure out how to import all the words from the dictionary text file into an array into my program that would allow me to compare it against my own user input array for possible matches. The hardest part that I was faced with was figuring out how to possibly find all the possible combinations of characters within each word. I had to refresh my knowledge on how factorials work as well as recursive algorithms. Afterwards finding any matches within the dictionary was a matter of nested for loops and I ran out of time before I was able to properly implement the function that would generate new anagrams for the found words due to my lack of time management skills and will proceed to do better next time.

I primarily stuck with using unbounded_string arrays that were technically uninitialized as they didn't have a static length. The unbounded_string arrays were used to hold all the dictionary words considering there are over 50,000+ words in the text file that we were supposed to grab from so attempting to predefine it's size would've resulted in a lot of potentially wasted memory and allocation of resources otherwise. I initially tried to use standard string arrays to hold dictionary words and the inputed words but after finding one too

many issues with that, I decided to change to unbounded string structures as they were somewhat easier to deal with.

From my experience writing the program in terms of attempting to find anagrams from user inputted strings, I would argue that Ada is definitely not well suited at all in regards to solving this problem as Ada tends to put a heavy emphasis on having initialized variables and static sized arrays which was constantly causing issues within the program as the dictionary file was absolutely huge which would consistently cause errors and crashes in terms of not having enough resources. There are unbounded strings which seemed like a hot fix of a dynamically allocated array but even then I ran into a lot of issues when doing literally anything with them.

Unbounded string arrays made Ada a good choice as they have the ability to be undefined lengths that interpreted strings as actual strings instead of just arrays of characters and checking characters within the strings as I had to numerous times throughout the word jumble assignment a lot easier to understand.

The benefits of Ada are the fact that it actually handles strings as strings instead of arrays of characters like C and other languages do which made dealing with string related problems a lot easier to handle and fix. For loops seemed absolutely confusing to use as it seemed that I needed to start for loops at 2 instead of 0 and I'm pretty sure that arrays start with 1 instead of 0 like most other languages which proved to be a minor annoyance as I'd constantly be trying to create loops and arrays with 0 as the beginning and I'd spend wasted time trying to figure out why arrays or strings wouldn't process properly.