Intro:

The main idea of this lab is to determine how to properly find the time complexities of multiple algorithms. We do this so that in the future, we will be able to determine the time complexity of various different algorithms and use that information for other things, such as determining if the algorithm is generalized for n items.

Implementation:

<https://github.com/avita2003/COM210_IntroLab.git>

The way I did this was by creating three different algorithms. The task of the first algorithm is to write a Java program to accept the names of three items along with their prices from the user and output them and the average price to a user. The first thing I did was create two arrays: Foods and Prices. I then used a scanner to allow a user to input the name of the items and the prices. Afterwards, I found the average of the prices, and outputted both the items and the average price.

The task of the second algorithm was to write a Java program to accept three item names, and prices, and output them, and to also output the average price if one of the items is named Peas (not case sensitive) otherwise output: “no average output”. I did everything the same as the previous algorithm, except I added an if loop to determine if anything in the array was “peas”. If the word “peas” was found, then the average price would be outputted, but if it wasn’t found, then “no average output” would be outputted instead.

For the final algorithm, I was tasked with writing a Java program to accept a given number of item names and prices and then output them in the reverse order in which they were input. In addition, output the average price if one of the items is named Peas (not case sensitive) otherwise output: “no average output”. I first started off by asking the user how many items they had, and then saved that number to an integer called Length. I then used that variable as the length of the arrays. I once again asked for the names and prices of the items, but then I used a StringBuilder to output the names of the items in reverse order. I once again found the average of the prices, and then used a for loop and nested if loop to determine if “peas” was in the array.

Analysis:

Time complexity is the amount of time it takes for a computer to run an algorithm. I will assign certain parts of my code to certain variables. Assignments will be C1, Comparisons will be C2, and Outputs will be C3. After coming up with these variables, I will go through the algorithms and assign these variables to the statements in my code. The time complexity for the first algorithm would be 3[C1]+Foods.length[C1+C3]+Prices.length[C1+C3]+2[C1]+2[C3].

The time complexity for the second algorithm is 3[C1]+Foods.length[C1+C3]+Prices.length[C1+C3]+2[C1]+C3+4[C2+C3]. Finally, the time complexity for the third algorithm is 2[C1]+C3+6[C1]+Foods.length[C1+C3]+Prices[C1+C3]+Foods.length[C1]+C1+C3+ Prices[C1]+C1+Prices[2(C2+C3)].

Conclusions:

In conclusion, time complexity is important because it allows us to determine how long it takes for an algorithm to run. One thing that is beneficial about time complexity is that it allows us to look at code in a different way. It shows us the basic structure of the code, which could allow us to have an easier understanding of how an algorithm works. Another benefit is that it could allow us to have an easier time changing the code to make it run faster. For example, instead of having two or three assignments, you could combine them into one big assignment. Or maybe you have an unnecessary comparison in your code. Being able to see your code in a more general sense may make it easier to spot these mistakes.