Stuart Wyse

CSE 465 – Term Project

Language Analysis

Table of Contents

# Zipcodes

C#

Python

# Summation

Java

Prolog

Scheme

# Z+-

Java

C++

# Band Matrix

C++

Python

**Zipcodes**

C#

This language allowed me to solve the zipcode problem(s) fairly well. Given that I don’t know C# very well, the main difficulty with this problem was just learning C#. Once I did that, I realized that C# had the features I needed to properly solve this problem. However, now that I have some experience with C#, I am confident that I would be able to solve problems more quickly with it in the future. I will absolutely still need a little refresher on C#, but I would hopefully be able to do that much more quickly. The run-time performance of C# was not noticeable different than Python in this problem. Overall, I would say the language is a reasonable choice, for me, with this problem. It’s reasonable given that I was able to solve the problem with C#, but did it more quickly with Python. I was able to solve this more quickly with Python mainly because I know and understand Python better.

Python

With Python, I was able to solve this problem without much of a hassle. However, there seems to be an infinite loop in my CityStates problem that I was unable to solve. I do not understand why or how an infinite loop is possible in it, but there is one in there. Even with that, the output file is still populated correctly. Python provided better features for this problem, since it allows for much easier reading & writing to files. The ability to read in a file and strip the data line by line was very helpful. I also thought that being able to split a line was helpful when searching for certain elements of the data. For the reasons just listed, I would say I noticed an appreciable difference with Python over C# when solving this problem. I already had a fairly solid background in Python, so there wasn’t as much of a learning curve as there was with C#. That being said, I think that I would be able to solve problems slightly quicker with Python in the future after solving this problem. As mentioned in the C# analysis, there was no noticeable run-time difference between the two languages. I would say that Python is the ideal language for this problem. The great write ability Python has when it comes to reading in and splitting up data from a file makes it the ideal language for this problem. It is especially easier than C# when it comes to reading in a file, as I used a StreamReader in C#.

Summation

Java

I was able to solve the summation problem using Java. Java had the features I needed to solve the problem. Essentially, I just needed something to store the list of values in. There was a significant difference in development time with Java over Scheme or Prolog when solving this problem. A huge part of that is simply because Java is the language I know and understand the best. For this reason, there wasn’t much of a learning curve with Java. When it comes to the run-time speed, I would say Java’s was slightly slower than Scheme. By no means was it *inefficient* though. Although the run-time was slightly slower, I would say Java is the ideal language (of the three used to solve this problem) for me. With Scheme and Prolog, there was a large learning curve that I had to get past before I could even begin. As I went on, I had to continually learn how to implement the different aspects necessary to solve the problem. With Java, I just had to figure out the best way to solve the problem. I didn’t have to research much about how to actually implement things. For example, in Java I could easily write code that would sum up a list. In Scheme, this was more difficult for me to figure out and I had to do a little bit of research from classes earlier this semester.

Prolog

Scheme

I came close to fully solving the summation problem using Scheme, however it was not 100% correct. It currently returns true if the target is in the list, or if the target can be reached by summing up n numbers from left to right. By that I mean that (1, 2, 3, 4, 5) with a target of 10 would be true. But (1, 2, 3, 5, 4) with a target of 10 would NOT be true, even though it should be true. The reason it doesn’t return the correct answer is because the numbers are only summed up from left to right, and 1+2+3+5 = 11. It definitely took me a little longer to solve this problem in Scheme than it did with Java, given that there was more of a learning curve with Scheme. If I were to do similar problems in the future, I’m sure I would be able to do so a little quicker. I would absolutely still need to learn and research more, because Scheme is a language I have a tougher time understanding. Scheme’s run-time to solve these summation problems was rather quick, and slightly faster than Java. Since its run-time is so fast, Scheme is a good language for this problem. However, since I am less competent with Scheme, I would say that it is reasonable. If I knew Scheme better, it would most likely be the ideal language.

Z+-

Java

I was able to solve the Z+- problem using Java. I made some fairly significant changes from the original Z+- homework to completely solve the problem. I also did not have any FOR loops working in the original homework, but I was able to implement that for the project. Java provided everything I needed to solve this problem. I used Java’s HashMap to store variables and their values and to update them properly, and I just used a Scanner to read in the Z+- file. I would definitely be able to solve this problem or similar problems more quickly in the future. Even though I know Java well, I wasn’t sure of what data structure I should use when I first started. Originally, I did not use a HashMap. So it took some time to implement that part of the solution. I did not notice any issue with the run-time of Java to solve this problem. Of Java and C++, Java is by far the ideal language for me to use to solve this problem. I understand and I am able to write Java much more quickly and efficiently than C++.

C++

I have most of this problem solved using C++. Really the only part I was unable to figure out was FOR loops. This took me some time to figure out in Java, and I unfortunately could not get it working in C++. There is also a Segmentation Fault occurring when I run the test script. C++ has everything I need to solve this problem. The development time was definitely slower than the time it took me to develop a solution in Java. Although my development time was slower in C++, I would be able to develop slightly more quickly in the future with similar problems. This is because I was able to work through a number of issues while solving this problem, and I would hopefully be able to avoid those issues next time. The run-time for this problem in C++ was good. I didn’t notice a significant difference in the run-times of the Java and C++ programs. Overall, I would say C++ is a reasonable language to solve this problem. As I mentioned in the Java section, I am able to write Java code more easily than C++, making Java my ideal language for this problem. That being said though, C++ can still be used to come up with a solution.

Band Matrix

C++

I was able to solve most of the Band Matrix using C++, but it is not fully complete. I get some errors when running the test script with my solution as well. C++ provided all of the necessary features I needed to implement a solution. I just do not know C++ well enough to actually use all of those features, unfortunately. It took me a while to develop the solution I have. I do not know if I could solve this problem much quicker in the future, as I found it very challenging for me. It is difficult for me to talk much about the run-time, since there are errors when I run the program. C++ is definitely a poor choice for me with this problem. Overall, I had a very difficult time working with C++ on this problem. A large part of that is because I am used to Java, where I do not need to worry about memory management as much. So, when it comes to C/C++, I have a difficult time working with pointers, addresses, and memory management.

Python