Emma Perez

Textual Spreadsheet with Ruby

Ruby is a dynamic language but is also object oriented. Everything in Ruby is an object, even numbers and primitive types. This makes it extremely flexible. Ruby doesn’t use curly braces, it simply uses blocks with “end” statements. I found that this makes it a little hard to read sometimes but if you indent everything correctly, there shouldn’t be an issue. I felt that Ruby was a good transition from Java. Since Java is object oriented, this concept was very easy for me to grasp in Ruby. Many things can be achieved with Ruby using less lines of code than Java. One very cool thing about Ruby is that you do not need to declare types. If you have a variable called “seven” and you want to assign it the value ‘7’ you simply say “seven = 7” with no type declaration. When defining arrays you say “arr = []” and hashes “hash = Hash.new().” This helped but also became a problem when I had to know the type of some variable in a hash or array. Ruby has a garbage collector and exception handling. The syntax is similar to Perl and Python. I chose to use Ruby because I have heard many good things about it. There are many jobs that require Ruby knowledge. This was a good opportunity to explore it and get some practice in. I felt that this would be the perfect language to use for this type of assignment.

For this assignment, we are treating a text file as a spreadsheet, essentially. Our group decided to use pipes as delimiters (‘|’). For the sake of other members we put a space at the start of a new line, and no delimiter at the beginning and end of a line. The start of a function is denoted by “[“ and the end with “].” This again was for the sake of other group members. I was able to just search for operators in the “cells” to look for a function. To read this into Ruby I created a for loop to read each line. I thought it would be best to break the whole file up and assign each ‘cell’ an appropriate column and row number. I used hash functions to create cell names and assign each of them row and column numbers. Each line that was read in was designated a row number, then I looped through that line and assigned it a number value based on a counter for each column number. The hash function stores the row and column number as the key and the cell value as the hash value, like this: “C1 R1 => 3”. So when it came time to evaluate, I simply searched for a key that matched the cell reference in the function and passed the keys’ value. For each product, sum, difference, and division, I created functions or methods for each one. In these methods, I would search for the appropriate operator, then split the function up by the cell references and search for the values. I added the values to an array and then multiplied the contents of that array. I had an issue with this at first when there were multiple functions with the same operator. It would add all the cells and then apply that operator to the whole array instead of just two at a time. So I after the second number had been added to the array, I would apply the operator, reassign the hash value, and then clear the array. I had trouble reassigning the hash value because I didn’t realize that the value had to be of type ‘hash’ for me to reassign a value. This required me to create a new hash function and temporarily assign it the value. Another issue that I had was duplicate values. Hash functions are meant to reassign a key value if there are duplicates. So all duplicates would point to the same cell or hash key. This became a problem when a function was pointing to a number that had been reassigned a new hash key. It took me a while to realize this was happening since my first text file didn’t have any duplicates. There was a simple way to fix this by calling the function “compare\_by\_identity” to the hashes. After the value is evaluated and reassigned the hash is made into an array so that it can be printed out. I create a new file called “evaluated.txt” and print each cell and its value to it.

I think this was a great chance for me to get introduced to Ruby. Any issues that I encountered were issues that could have been avoided had I known the syntax more. I didn’t find anything that could not be done in Ruby for this project. This seemed to be a good introduction to functional/dynamic languages, especially having knowledge of Java. During the entirety of this project I was also doing a group project for Software Engineering which required me to be coding in Java. I found that I could easily be coding in Ruby and then pick up where I left off in Java and have no issues. The syntax isn’t identical but it is comparable. Overall, I enjoyed Ruby and I hope that I use it in the future.