Report: HW Ruby Part 1 and HW Ruby Part 2

Amin Babar

10/04/18

CPSCI – 220

In order to start these problems, I had to look into how arrays and blocks are handled in Ruby. I had to look for the methods pertaining arrays on ruby docs. After getting used to arrays, I looked at blocks and how to pass parameters to them. I specifically looked at the map, each and select methods that take code blocks. I found these higher order functions really helpful. I did not have to worry for the most part about keeping track of the indexes because I could simply call one of these methods on the input file. Here's an example of how this helped me write succinct code in assignment 2:

```
legal_words = dictionary.map {|x| x.chomp}
@legal_words = legal_words.select do |x|
        x.length > 2 and x.length < 7
end
```

I was not that familiar with code blocks for assignment 1, but by the time, I was working on assignment 2, I was more familiar with code blocks, and that made my life a lot easier.

I implemented problem 1 using arrays, but that was really slow, so I ended up using hash which is much more suited for assignment 1. The keys for the hash were all the consecutive 2 worded strings from the input file, and the value for the hash was an array of all the words that followed the hash. The .sample method on arrays helped with generating random elements which was really helpful for creating trigrams. I also ended up using regular expressions for

assignment 1 which were really helpful in generating sentences by recognizing capital letters for the start of the sentence, and end of sentence punctuation for the end. By defining the each function for assignment 1 and by including the enumerable class, I got access to numerous other methods on my class objects which was convenient.

For assignment 2, I had to handle the time out exception. Thankfully, Ruby had a simple solution for that. I found the concept of using variables within a string simple in ruby. Unlike python, it was easy for me to include variables in a string and output them. Eg:

```
puts "Total words guessed:         #{@total_words_guessed}"
```

This helped in producing the rightly formatted output.

I found the concept of there being multiple methods with the same purpose but different names annoying because all the available options would end up confusing me. Also, the new line character that is produced after puts made formatting slightly difficult, but I was able to overcome that by using the combination of print and puts.

Overall, I found ruby to be a convenient language for these assignments. The features of arrays, hashes, code blocks and the ability to declare variables without their type was really helpful.