CSC 3370 – Ruby Project

The purpose of this assignment is to give you experience building simple programs in Ruby.

Part 1 - Simple Counts

For this part of the project, you will write a simple Ruby script that will calculate word counts for a plain text. The goal for this project is to write clean, concise, and readable code for text processing; Ruby is an excellent language for this task.

Create a simple text file called test.txt with some test words in it. You may use any editor you wish, but it must be capable of editing plain text files.

Here is a an example of some text that you could use for testing:

```
test text

here is a test

more text
```

Create a Ruby script called **word_count.rb** in the same folder. In that file, write a method called **count_words** that takes a single filename as a parameter and returns a Ruby Hash where the keys are words from the file and the values are the counts of those words in the file. Here is some pseudocode for that method:

```
create new hash table
for each line in the file:
  parse the line into words
  for each word:
    increment the corresponding count in the hash table
  end
end
return hash table
```

Hints:

- Check out the Ruby documentation for Array, Hash, and String
- You may wish to choose a default value for the Hash object. You can do this by passing the default value as a parameter to the Hash constructor.
- Recall that you can iterate over all lines in a file using the following syntax:

```
File.foreach(filename) do |line|
 # process line
end
```

- Recall that you can split a Ruby string on spaces (or any other pattern) using the split method.
- Write some code outside the function at the bottom of the file that calls the function with the filename of the file you created earlier and stores the result in a variable. Then iterate over the word-count pairs in the resulting Hash and print them out. Here are sample results:

  test: 2
  text: 2
  here: 1
  is: 1
  a: 1
  more: 1

Part 2 - Nicer Counts

Now that the basic word counting routine is working, switch to working with a larger input text. Download a larger text from somewhere on the web.

If you run your old code on this, the results likely will not be terribly illuminating because it will generate a lot of output without any organization (and probably with some incorrectly-detected "words"). You need to make several improvements to your script:

- Improve your parsing routine by using a regular expression for the splitting. The regular expression should match various punctuation symbols (commas, colons, etc.) as well as whitespace characters as potential word breaks.
- Make the counting case-insensitive by converting words to lower case before storing them in the Hash.
- Filter the results by only printing words that occur more than five times in the source document.
- Improve the relevance of the results by sorting them so that the most frequently-seen words are at the top (or bottom) of the list. You may wish to look into the sort_by method in the Hash class.

Here is an excerpt from some example results after improvements have been made:

```
...
congress: 60
have: 64
as: 64
any: 79
state: 79
united: 85
for: 85
a: 97
by: 101
president: 109
states: 129
```

```
in: 147
...
```

Grading Rubric

- Part 1 implemented completely and correctly (70 points)
- Convert words to lower case correctly (10 points)
- Regex for splitting words (10 points)
- Filter words (5 points)
- Sort words (5 points)

Submission

You must upload a copy of your work through the portal. If you have any issues or questions please let me know.