**CS 1632 – DELIVERABLE 2: Unit Testing Ruby Rush**

**William Stephenson**

**willgs**

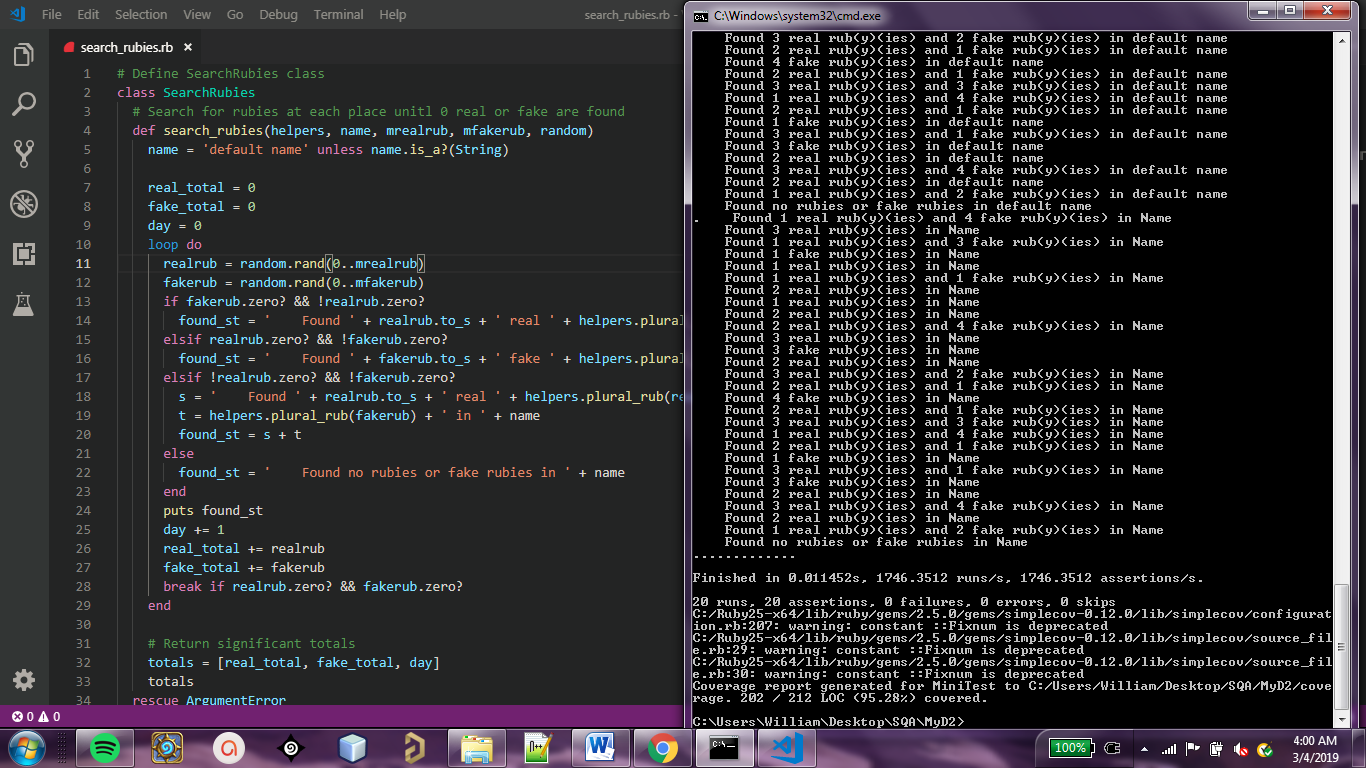
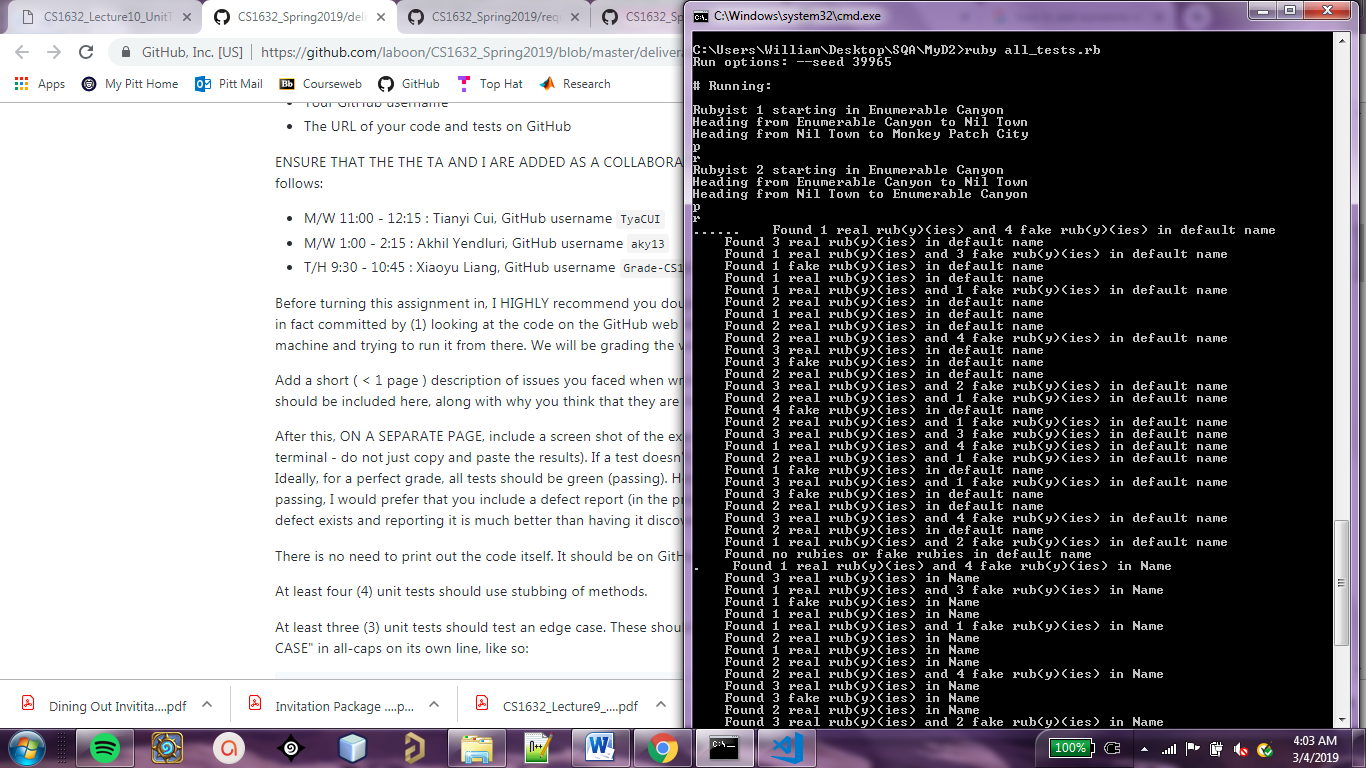
[**https://github.com/willgs/D2.git**](https://github.com/willgs/D2.git)

**Issues Faced**

On the surface the requirements for the code itself were not overly challenging. I was able to meet each requirement listed as I went along with little trouble. One of the most challenging parts of coding this project was dodging the Robocop. There are several spots in my code that work fine but look very odd; these are where I had to evade the Rubocop. Besides evading this merciless ruby law enforcement officer, I had very few issues accomplishing what I needed to and debugging along the way.

Once I got to the point of actually writing test plans, however, I realized I should have paid more attention to the way I was setting up my code in terms of testability. When I first considered the code ‘finished’ and attempted to write some tests, I had a lot of trouble. Several of my methods were as large as Rubocop would permit, almost everything relied on everything else, and it was all in only two classed. Smaller methods and class separation proved to be very important and, after a few rounds of restructuring, I was able to run reasonable unit tests.

Writing tests for success\failure methods seemed to come a little bit more natural to me than the equivalence class methods. I believe that is just because I naturally view most pieces of code as either not working or working, and I can easily see when I why a method would move from working to not working and back again. I was still able to write about the same number of equivalence class method tests, after I put enough thought into it. I had to put a little bit more effort in recognizing what an equivalence class would be for a given method and what would be a reasonable result to expect from each class.

**Screenshot of Executed Unit Tests**