**Summary and Concerns**

For this deliverable, the main concern was to build our first Ruby program. We intended to use the Red-Blue-Green principle in development however with limited knowledge of Ruby that process could have taken a lot more time. Instead we approached this deliverable by splitting the main program versus the unit tests in a code, test, refactor way.

The focus when developing the program was to implement small, pure methods such that the method could easily be associated with a test. For example, the program had a lot of reused phrases so instead of using a put or print they were grouped into a method that displayed the correct phrase every time. Implementing the program in this way benefitted us in 2 ways. Grouping similar functions allowed for easier testing because a method could be tested just once, and it also helped with debugging the code. There is some concern about missing rescue functions in the current implementation however testing could be used to identify some of the areas where a rescue was not implemented.

In regard to testing there were some concerns during creating unit tests as well such as what defined a “trivial” unit test and how many tests should be sufficient to meet the requirements but not get points deducted. For example the srand() function is important so that the same number can be seeded every time and the program can be repeatable. If the arguments are checked when the program is initialized should the equivalence class for non-integers still be added to the unit test for srand?

Not using the Red-Blue-Green method came be an annoyance later. When trying to test methods sometimes the remainder of the program would run. An example rose when we were testing arguments. It took a bit of thought and rereading to figure out that checking the arguments in a different class would stop the main program from executing. Overall by using small pure methods, it was easier to refactor the code in such a way that these problems could be fixed.