**Some notable obstacles I overcame:**

This project was definitely challenging. After reading the spec, I took a while just conceiving the design of the code I was going to implement. I attempted trying out the manageOneRound method first in order not to have to deal with random words from the wordList but I was stumped for a decent amount of time trying to figure out how to get the pebbles to work. Initially, I had mistakenly presumed pebbles were to be counted for every repetition of a character but after carefully reading the spec, I realized a character of the pebble could be counted as a pebble only once. After hours of thought, the pseudocode I came up with involved two integer arrays, one called rockArray that would take all positions where a letter of a probe and the respective letter of the secret word at that position were the same, and a pebbleArray that would take in all the positions of the rockArray and any additional positions of letters that match between probe and secret word, regardless of position. Looping through the probe, an iteration would be skipped when a position matched any entry within the rockArray. Within the probe loop would be an inner loop that would skip an iteration every time a position matched any entry of the pebbleArray. Another obstacle I faced was trying to keep track of the counters of the two different arrays as I was using while loops for both of the secret and probe loops.

**Pseudocode:**

Design of main:

* Display output messages
* Call manageOneRound for number of rounds inputted
* Calculate average, min, max values
* Display results

Design of manageOneRound:

* Repeat until the probe matches the secret word
  + Prompt for a probe
  + Check for invalidities (invalid length, if it is a word within the wordList)
  + Check if probe matches secret word and terminate if it does (returning no. of tries +1)
    - Search for rocks using a single position counter
      * Go through both probe and secret word
      * Store matching positions in 1st array, these positions will be skipped
    - A 2nd array will be created with positions of rocks and pebbles too
    - Search for pebbles
      * Loop through probe for matching positions with 1st array entries and skip an iteration when matched

Loop through secret and move to next iteration of this inner loop if any match an element from the 2nd array

* + - Output number of rocks and pebbles
* Default return value should never be reached (returns -1)