2a. One notable obstacle I had to overcome was breaking down the playOneRound function. While it seemed like a lot to tackle from the outside, it was much easier to take on when breaking the large function down into smaller pieces. Additionally, I had trouble correctly calculating the number of silvers versus golds. I had to use the debug console to find out my issue.

2b.

Main routine: In my main routine, I load the dictionary into the made array by calling the getWords function, then placing a check to ensure the words actually loaded. I then continue to prompt the user for the number of rounds to play, and then initialize my min, max and accumulation variables. The accumulation variable is to tally up total amount of attempts for each round to calculate the given average. I call playOneRound and it returns the number of tries for that given round. I then correctly set the min, max and accumulation variables, and print out the statistics.

playOneRound function: Makes sure wordnum is a positive number in the right boundaries, prompts user for probe word, runs isValidWord and isInDictionary in a while true loop to continue prompting user until correctly guessed hidden word, and makes sure input is valid. Input check to be between 4-6 characters, and is a real word. Another while loop inside while true loop to keep prompting the user for valid input and printing the respective error message based on users input. Once given valid input, increment counter variable, and call numGolds and numSilvers to calculate number of gold and silvers. Checks if probe word exactly matches random word, if not, prints number of golds and silvers and prompts user for probe word again.

isValidWord: iterates through probe word to ensure all letters are lowercase, checks the length of probe word to make sure between 4 and 6, if any of these conditions true, it will return early, otherwise returns at the bottom

isInDictionary: iterates through entire dictionary until the user probe word is found/not found

numGolds: finds if length of probe word or hidden word is shorter to know how long to iterate over word, starts at first index of hidden word and iterates over the respective index of probe word to see if both letters are in the same spot, if they are, increment gold counter. Return number of golds found at the end.

numSilvers: stores user guess into temporary variable to alter that string and not original string, starts at 1st index of target word, and checks every letter of guess word to see if any letter matches. If there's a letter match, change that respective index in the given word to a ‘0’ char to make sure not double counted if repeated letters in hidden word. Returns the number of silvers found and subtracts gold from silver because letters are double counted.