## Obstacles Overcame

One of the obstacles I overcame, which is also the most notable one, is how configure a complete Xcode project involving multiple files. This took about half an hour before I even started writing the program, but I really appreciate the experience. In our past projects, usually there was only one main.cpp involved; but this project has a separate cpp file named utilities.cpp with its header file utilities.h, together with the word list file words.txt which is essential to the project. All these parts work together to complete a Wordle game. For the first time I get a taste of managing a project: what I create is now beyond a single file performing a repetitive task, or executing the same order, but really a whole package. In other words, this feels real.

Another obstacle I overcame was using <cstring>. String operations in C is less convenient than in C++, where operators like “+”, “=”, and “==”, and functions like str.length() are disabled, therefore I needed to learn how to use various functions in <cstring> like strcpy, strcmp, strcat, strlen, etc. This is quite different from C++ where a lot of things seem easier and more direct, but I did enjoy the learning experience.

Another obstacle I overcame was how to calculate the correct number of golds and silvers. Initially my method is to count gold whenever guess[i]==ans[i], else count silver for guess[i]==ans[j]. However, this results in over-counting: when I test the program using answer “**raven”**, guess **“never”** returns 2 golds and 3 silvers when it should have been 2 golds and 2 silvers. This is because both “e”s in “never” were given a silver when there is only one “e” in the answer word “raven”. Based on the rule that “neither of the two letters should be involved in a gold or another silver”, I rewrote this part to introduce a “temporary-ans”, then erase both letters in that string and the guess string whenever a match is made, to ensure no over-counting.

## Pseudocode and Description

**playOneRound function**

*if the integer parameters (nWords, wordnum) are invalid*

*return -1*

*read the probe word (the guess)*

*while the guess does not match the answer*

*if the length is not between 4 and 6*

*ask for correct length*

*if the string is not in lowercase letters*

*ask for lowercase*

*if the string is not in vocabulary*

*say I don’t know that word*

*the attempt counts as a try ++*

*scan through the guess string*

*give gold if the letter matches exactly as in the answer*

*mark both letters as used*

*scan through the guess string*

*scan through the answer string*

*give silver if the letter matches anywhere in the answer*

*mark both letters as used*

*print gold and silver summary*

*read a new probe word (new guess)*

*return the number of tries*

Two other functions checkLower and checkVocab were written to test whether the string is in lowercases and whether the string is a valid word. The two functions perform simple tasks with a loop that scans through the given arrays and determines if conditions are satisfied.

Additionally, when giving golds and silvers, I change the used letters to irrelevant letters (upper case ‘X’ and ‘Y’ )to keep track of which letters are used, preventing over-counting.

**main function**

*declare the word array and other variables*

*load words.txt into array, report and terminate if n<1*

*read number of rounds, report and terminate if rnd<1*

*loop for the number of rounds*

*print which round*

*choose the hidden word using randomizer*

*print the length of the hidden word*

*playOneRound*

*record the number of tries (score) in this round*

*compare the score this round with max and min, update*

*add score this round to total score, calculate new avg*

*print the avg, max, min so far*

*return 0*

If it is the first round, then initialize max and min value with the score in the first round. Else, compare and update. Also, since function playOneRound is only responsible for the guessing and comparing part, the hidden word must be chosen and its length printed before calling the function; various score recorded and calculated only after the function returns.