**Project 5 Report**

Notable Obstacles:

The main obstacle I encountered in project 5 was coming up with the algorithm for determining the number of rocks and pebbles. My initial algorithm calculated the number of rocks correctly; however, words with two of the same letter would cause the letter to be double counted so that the number of pebbles would be incorrect. For example, if the secret word was detest and the probe word was detent, it would incorrectly return 5 rocks and 4 pebbles when it should be only 5 rocks and 0 pebbles. The letters e and t were being double counted when some of the e’s and t’s were already counted as rocks. I was troubled by this problem for hours, so I decided to stop working on the project for the day and resume my work tomorrow. And the next day I worked on it, I was able to come up with an algorithm that worked correctly. This incident reminded me that if I ever get stuck on solving a problem, I should take a break, refresh my mind, and try again.

Program Design:

int main( )

if number of loaded words < 1

print no words loaded

return 1

ask how many rounds user wants to play

if number of rounds negative

print number of rounds must be positive

return 1

for each round until all rounds completed

select secret word from list of words

print round and length of secret word

execute one round

calculate average, minimum score, and maximum score

print round score, average, minimum score, and maximum score

int manageOneRound(const char words[][7], int nWords, int wordnum)

if nWords is negative, or wordnum is less than 0 or wordnum >= nWords

return -1

while round is not complete

ask for probe word

if length of probe word is less than 4 or greater than 6 or letters not lower case

print probe word must have 4 to 6 lower case letters

restart round (the while loop)

if probe word is not a word loaded in the list of valid words

print that word is unknown

restart round (the while loop)

add one to the score

if probe word matches secret word

round is complete

break out of round (the while loop)

for each letter in probe word and secret word at the same position

if the letters are the same

add one to rocks

set that letter in both secret word and probe word to 0

for each character in probe word and secret word

if the letter in secret word or probe word is not 0 and the letters match

add one to pebbles

set that letter in both secret word and probe word to 0

print out number of rocks and pebbles for that probe word

return score of that round

int isNotLower(const char probeWord[200])

for each letter in probe word

if one letter is not lower

return that probe word is not all lower case

return that probe word is all lower case

int isWordValid(const char words[][7], char probeWord[200], int nWords)

for each word in the loaded list of valid words

if probe word does match of a valid word

return that word is valid

return word is not valid