1. Notable Obstacles

I found the countPebbles function especially tricky, for it requires you to mind many details . And if you don’t, it will sure pass the right answer for some cases, but fails the other cases which are the majority. I solved it by adding two arrays that can show whether this letter is already being used in a rocks pair or another pebbles pair, thus making the function do what is required in the spec.

2. Brief Description of My Project

**main function:**

call loadWords to fill the array

get nWords

if nWords less than 1,

return -1

Choose how many rounds to play

if the round number less than 1,

return -1

declare the times of guesses in each round

for every round

select random word as secret word

get the length of the secret word

play a round with that word

when the function manageOneRound returns -1,

return -1

special case when only need one guess,

cout one time

update and print statistics

cout average, minimum and maximum times of try

**manageOneRound function:**

Set two boolean

if wordnum beyond the range,

return -1

set probeNum equals to 1

if the first probe word equals to the secret number,

return 1

set the two boolean to false

check if probe word has a valid number of lower case letters

check if the probe word is in the word list

report number of matching letters by calling two functions

**contains function**

do k increment

see whether the probe words are in the wordlist given

if so,

return true

if not,

return false

**countRocks function**

set nRocksMatches to 0

do k increment

if s1[k] = s2[k],

do nRocksMatches increment

set used1[k] = true, used2[k] = true

return nRocksMatches

**countPebbles function**

set nPebblesMatches to 0

do k1 increment

do k2 increment

if s1[k] = s2[k],

do nPebblesMatches increment

set used1[k] = true, used2[k] = true

break to avoid repeated pebble pair

return nPebblesMatches