1. One of the most notable obstacles I faced while coding for this spec was finding the pebbles. I tried doing that by comparing the each word of the secret word to every letter of the probe words, which gave me issues like retaking the rocks and also extra pebbles when there were two same letters in either the probe or the secret word. I solved this by changing the letter in the probe word when it was identified to be equal.
2. manageOneRound

if nWords is negative and the number for secret word isn’t between 0 and nWords

return -1

*repeatedly*

temp c string “a” is made a copy of the secret word

incrementing number of rounds

taking in probe word from the user

temp c string “b” is made a copy of the probe word

-----------

*repeatedly*:

if all characters in the probe word are not lower case characters

incrementing temp variable chk

-------------

if chk is not 0 and number of characters aren’t between 4-6

print output required by spec

decrement number of rounds

continue to next iteration of outer loop

------------

*repeatedly*:

if word is not present in the array containing all words

-------------

print output required by spec

decrement number of rounds

continue to next iteration of outer loop

-----------

*repeatedly*

if each letter of the probe word is the same as each letter of the secret word at that position

change that letter in the probe word to character \* and that character in temp variable for secret word to +

increment rocks

--------------

*repeatedly*

if each letter of the secret word is the same as any letter of the secret word at any position

change that letter in the probe word to character \*

increment pebbles

break

-------------

if secret word is same as probe

break

print rocks and pebbles

---------

int main()

if words are loaded in the c string array

print output required by spec

end program

take in number of rounds

if rounds are non positive

print output by spec and end program

-----------

*Repeatedly:*

Print round number

Call rand function to select a random number between 0 and nWords-1

Print length of secret word

Call manageOneRound

If score is 1

Print output as required

Else

Print output as required

Store scores in a temporary integer array

Sum calculates sum of all scores till that round

Average calculates sum of all scores till that round divided by number of rounds

Calculate minimum and maximum by comparing each element

Print all these values

---------