1. Notable Obstacles:

* While outputing the average number of attempts it didn’t count the decimal places, despite being double. I used static\_cast<double> on both integers of the quotient to solve it.
* When counting pebbles, for the function no to count rocks as pebbles or multiple equal letter as multiple pebbles (ex: assert vs rose; counts 2 pebbles for ’s’). To solve this I created 2 bool arrays (1 for the random word, 1 for the player word) of MAXWORDLEN+1 size to store the state of each character, true if character can be used, false if it was taken by a rock or another pebble

b) Pseudocode:

* main function
* manageOneRound function
* goodWord function
* rocks function
* pebbles function

main function:

set cout decimal precision to 2

create an array to store the wordList and load the word list onto it

store in an int the actual number of words loaded

if the number of words loaded is <1 return -1

require input of number of rounds to play, if <1 return -1

create variables to store min number of attempts, max number of attempts,

and average number of attempts

repeat as many times as indicated by nRounds (player input)

output round number

select random number to choose word from the list

output the length of the random word selected

call manageOneRound

output number of attempts needed to succeed

if number of attempts is the lowest, replace minAttempts value

if number of attempts is the highest, replace maxAttempts value

calculate the average number of attempts

output: average, min attempts and max attempts

manageOneRound function:

if nWords or wordnum are <1 or wordnum >= nWords Return -1

create int variables for the function: nRocks, nPebbles and attempts(1)

create char array to store the players inputed word: pWord[]

create bool variable to determine continuity of second loop: theTruth

loop until determined to break

loop until determined to break (word input/check)

ask user to input a word

call goodWord to check if the word is acceptable

if the word is acceptable end the loop

call nRocks to determine nRocks to determine number of rocks

call nPebbles to determine number of pebbles

if the number of rocks is equal to the strlen, break the loop

output the number rocks and pebbles

increase “attempts” value by 1

Return the number of attempts

goodWord function:

check if inputed word has 4, 5 or 6 characters

if not, output message and return false

repeat for each letter in the string

if the character isn’t a lowercase letter, return false

check if the word matches any word on the list

if it finds a match, break the loop and return true

if no match is found, return false

rocks function:

repeat from 0 to the shortest strlen between the random word and the inputted word

if the characters from both words are equal

add 1 to the number of rocks

return the number of rocks

pebbles function:

create a bool array to store the state of each character in the random word (taken or not)

called nonValidR

create a bool array to store the state of each character in the inputed word (taken or not)

called nonValidP

set all values of nonValidR and nonValidP to true

repeat from 0 to the shortest strlen between both words

if rock is found, set the position of the character from nonValidR and nonValidP

as false (taken)

go through every letter of the random word

go through every letter of the player inputed word

if a letter from the random word and the inputed word match, and neither

of them have been taken (array positions are true)

increase number of pebbles by 1

set the value of the array of both characters to false (taken)

return number of pebbles