**A brief description of notable obstacles you overcame.**

It was difficult to start the project because I had to read all the instructions and make sure that I set up the files and the code correctly (e.g. putting in the loadWords function, linking a word.txt file, getting my head around what the variables represented)

It was difficult at first to make sure program returned and printed things that I wanted because sometimes I left out the square brackets after certain array variables and nothing printed and it was very confusing

It took me a while to think of how to set up the manageOneRound function because I had to make sure the invalid probe words would output the correct sentences and not add to the score but also make sure that the probe words would be checked first to see if they were valid before adding to the score. Finally came up with a while loop with two extra functions (probeWord4to6lower and wordPresent).

It was challenging to think of a way to make sure that pebble counting would not repeat on matches between the probe and the secret word that had already been counted, while also making sure that the rocks would not be included in the pebble count.

**A description of the design of your program. You should use pseudocode in this description where it clarifies the presentation. Document the design of your main routine and any functions you write.**

**bool probeWord4to6lower**

* if probe word is less than 4 or more than 6 characters
  + return false
* if probe word contains any characters that are not lower case letters
  + return false
* return true

**bool wordPresent**

* loop through every word in word list
* if a word in list is same as probe word
  + return true
* return false

**int manageOneRound**

* if number of words in list or random integer referring to secret word is negative or if random integer referring to secret word is larger or equal to number of words in list
  + return -1
* input probe word up to and including 99 characters
* declare score = 1 as int
* while probe word is not the same as secret word
  + if probeWord4to6lower is false
    - print “Your probe word must be a word of …”
  + else if wordPresent is false
    - print “I don’t know…”
  + else
    - add 1 to score
    - declare rock = 0 and pebble = 0 as int;
    - loop through letters in probe word and secret word until reach end of shorter word
      * compare each letter sequentially, if same letter, add 1 to rock
    - loop through secret word until reach end of secret word
      * declare secretLetter = 0 (number of probeWord[k] letters in secret word) as int
      * declare probeLetter = 0 (number of probeWord[k] letters in probe word) as int
      * declare multipleSecretLetter (if there are any probeWord[k] letters in secret word coming before position k)
      * loop through letters coming before k position in secret word
        + if probeWord[k] letter appears before k position, return true
      * if multipleSecretLetter is false (the letter is appearing for the first time in the secret word)
        + count number of probeWord[k] letters in secret word, add to secretLetter
        + count number of probeWord[k] letters in probe word, add to probeLetter
        + compare secretLetter and probeLetter, add lesser or either if both are equal to pebble count (this counts number of matching letters in both words, regardless of position, i.e. it also includes rocks)
    - deduct rock count from pebble count
    - print “Rocks: rock count, …”
  + input next probe word up to and including 99 characters
    - if word is right, leave while loop
* return score

int main()

* declare word list as c string array
* declare nWords as int (number of words in word list)
* if nWords < 1
  + print “No words were loaded…”
  + return 0
* declare rounds as int
* print “How many rounds…”
* input number of rounds
* if rounds ≤ 0
  + print “The number of rounds must be positive.”
  + return 0
* loop for number of rounds entered
  + print “Round …”
  + declare wordnum (random integer between and inclusive of 0 and nWords-1) as int
  + print “The secret word is … letters long.”
  + print score from manageOneRound function
  + if score is 1, print “You got it in 1 try.”
  + If score > 1, print “You got it in … tries.”
  + declare average as double
  + declare minimum and maximum as int
  + if it is first round
    - minimum and maximum will equal to score
  + calculate average
  + if score is less than minimum, new minimum will equal to that score
  + if score is more than maximum, new maximum will equal to that score
  + set precision of double to two decimal places
  + print “Average:…”