Report

1. I had difficulty accounting for probewords and secret words that had double same letter repeated. I realized that “blacking out” the letter with a newline character would cover this situation. I had to create temporary versions of the secret word and the probe word, so the original versions would still be checked if equal to each other whenever entering the checking portion of the while loop. I had difficulty accounting for whether a probeword was a valid probeword. I realized creating separate bools that would change once a character was invalid would reveal the answer.
2. Include libraries

main function

Load words

Check if number of words in list is valid

Prompt for rounds

Check if rounds are valid

Set up min, max, and average

For loop to run though rounds

Choose random word

Reveal how many letters in secret word

Catch attempts into manageOneRound

update attempts output

sort out statistics and output them

manageOneRound function

initialize probeword and attempts

do while loop begins

get probeword

check if probeword is within boundaries, 4-6 lowercase letters and within list of words

create temporary versions of probeword and secret word for manipulation

for loop runs through temporary probeword and temporary secretword

checks for rocks: matching aligning characters

blacks out used characters

for loop runs through temporary probeword

if letter has been blacked out, continue for loop

for loop runs through temporary secret word

checks for pebbles: current elements in both words match but are not in the same position

blacks out used characters

count up attempts

check if probeword and secret word aren’t equal

outputs rocks and pebbles

do while loop ends when probeword and secret word are equal

return attempts