Nathan Tsai

Smallberg

CS 31

14 November 2015

**Notable obstacles:**

How do you count pebbles correctly without over-counting? After I thought I completed the countPebbles method, I realized that it over-counted a lot of letters. For example, if the probe word had three ‘e’s, but the secret word only had one ‘e’, it would count three pebbles for ‘e’, even though there only exists one pebble. I had to write code that took into account that it must avoid repeated letters like that, unless the secret word has multiple letters as well. I also had to be creative with my validate method, and make sure that the correct error message displayed at the right time. I had to make sure that the “4 to 6 lower case letter” message was prioritized over the “I cannot find this word” message. The second message listed must only be displayed if it passes the “4 to 6 lower case letter” test. After overcoming these obstacles, everything else went smoothly.

**Description of program design using pseudocode:**

Method that counts letters in a given c string

Loops through each character of c string, count++ whenever it counts a character

Return the count number

Method that counts Rocks given two c strings (probe word, secret word)

Loops through each character of secret word

If the same letter shows up in the same position, count++

Return the count number

Method that counts the number of pebbles given two c strings (probe word, secret word)

Loops through each letter of the secret word

Variables that count how many times a letter shows up

Checks if same letter appears twice in the secret word

If no repeated letters, start counting pebbles

Counts # times a specific letter shows up in secret word

Counts # times a specific letter in the secret word shows up in probe word

Compares the counts, adds smaller number to total pebble count

Returns total number of pebbles minus the number of rocks

Method that checks if probe word is between 4 to 6 lower case letters, and checks if the word exists in the wordlist

Two Booleans that represent each condition listed above

If length is between 4 and 6, and if casing is correct, set first bool equal to true

If it isn’t syntactically correct, cout an error message about the letters

If first bool equals true, checks if word exists in the wordlist

If it doesn’t show up in list, cout error message saying game does not know word

Method that manages one round

If number of words in the array is not positive, or if wordnum is less than zero or greater than or equal to nWords, return -1

Loops until correct word is found

Prints “Probe word: “

Takes in user input

Calls validate function to make sure probe word is valid

If the probe word equals secret word, leave loop

If the word is valid, print out number of rocks and pebbles

Number of tries is recorded and numTries++ after each valid try

Exits Loop when correct word is found

Returns the number of tries

Main Method

Checks if word list is empty, exits program if empty word list

Prompts user to enter number of rounds

If number of rounds is negative, cout error message and exit program

Do while loop that calls manage round function

Choose a random number

Cout Round number and how many letters secret word is

Store number of tries from manage one round function

Total score is number of tries added for each round

Calculates average with totalscore/roundnumber

Couts different success messages based on number of tries

Sets a min and max variable, that updates after every round

After every round, cout average, minimum, and maximum score

Increases round number at the end of the loop

Stops the loop after round number equals the number of rounds that user inputs