**CS31 Project5 Report**

**Xuanhe Zhang (706078980)**

**Question 1: A brief description of notable obstacles you overcame.**

1. How to calculate gold and silver correctly is the hard part of the program. I use two int arrays *bin1[]* and *bin2[]* to record the number of occurrences of each character in each of the two words. I use the *character – ‘a’* (The difference of their ASCII Code) to get its position in two bins.

First, we iterate through the answer word. If the index is less than the length of the guess one and the character on that position is as same as the guess word, we add 1 to gold. Then, similarly, we iterate through the guess word. If the index is less than the length of the answer word and the character on that position is as same as the guess word, we continue. (Since we already count that in gold) Finally, we iterate through bin1[] and bin2[], and add the smaller value of each character in the corresponding position of two bins to the silver.

2. I learned how to include other files in the main C++ file and how to run a program split into several files.

3. Since we can’t use the string library, I have to find the corresponding C string methods to get, compare, and copy strings.

4. The hint about the word length was initially placed in the PlayOneRound() method, but it should have been placed in main().

**Question 2: A description of the design of your program.**

The judgeWord() method which count the gold and silver in words is the core algorithm of the program.

I also used some helper method that return boolean or print out sentences to keep the structure of the program clear.

int **playOneRound**(const char words[][MAXWORDLEN + 1], int nWords, int wordnum);

//play one round

void **judgeWord**(char[], char[]);

//compare the guess word with the answer and calculate gold and silver

bool **isValidInput**(char[]);

//check if the input is valid

bool **containsWord**(const char words[][MAXWORDLEN + 1], char word[MAXWORDLEN + 1], int nWords); //check if our dictionary contains the guess word

void **printStatus**(int, int);

//helper method to print the status of the golds and silvers in each guess

void **printResult**(int, int, int, int, int);

//helper method to print out the result at the end of each round

void **printInvalidInput**();

//print the invalid input warning

void **printNotContains**();

//print the don’t know that word warning

void **printNotLoaded**();

//print the no words were loaded warning

int **main**() {

get **nWords** by calling **getWords()**;

if **nWords** is less than 1, call **printNotLoaded()** to print out warning and return 1;

print out prompt sentence to let user type in;

get the **playRounds** from user input;

if **playRounds** less than one, print out the must be positive warning and return 2;

loop **playRounds** times:

print out round number;

get a random number from **randInt()**;

print out hidden word’s length;

call and get the **score** with **playOneRound()**;

compare and preserve the **minScore** and **maxScore**;

add **score** to **totalScore**;

call **printResult()** to print the result of one round;

}

int **playOneRound**(const char words[][MAXWORDLEN + 1], int nWords, int wordnum) {

if **nWords** less than 1 or **wordnum** (random number) excessed the boundary return -1;

set **score** with 1;

infinite loop:

get the **guess** word from user input;

if **guess** word is not valid, call **printInvalidInput()** and print warning;

else if dictionary doesn’t contain **guess** word, call **printNotContains()** and print warning;

else if **guess** word is not equal with the **answer**:

**score** add 1;

call **judgeWord()** to count gold and silver;

else break;

return **score**;

}

void **judgeWord**(char answer[], char guess[]) {

using **bin1[26]** and **bin2[26]** to store the number of the characters in two words;

loop through the **answer** word:

if current position less than **guess** word’s length and characters on that position in **answer** and **guess** are same, **gold** add 1;

else the number of characters which stores in corresponding position in **bin1[]** add 1;

loop through the **guess** word:

if current position less than **answer** word’s length and characters on that position in **answer** and **guess** are same, continue;

else the number of characters which stores in corresponding position in **bin2[]** add 1;

loop through the 26 positions in **bin1[]** and **bin2[]**:

**silver** add the minimum number of characters in **bin1[]** and **bin2[]**;

call **printStatus()** to print out the numbers of **gold** and **silver**;

}

bool **isValidInput**(char guess[]) {

if **guess** word’s length is out of the range, return false;

loop through **guess**:

if character in **guess** is not in lower case, return false;

return true;

}

bool **containsWord**(const char words[][MAXWORDLEN + 1], char word[MAXWORDLEN + 1], int nWords) {

loop through **words**:

if find **word** in **words**, return true;

return false;

}

void **printStatus**(int gold, int silver) {

print out the status of **gold** and **silver**;

}

void **printInvalidInput**() {

print out the invalid input warning;

}

void **printResult**(int score, int totalScore, int round, int min, int max) {

if score equals 1, print out singular result;

else print out plural result;

set the precision of output;

calculate and print out the average score, the minimum score and the maximum score;

}

void **printNotContains()** {

print out the don’t know that word warning;

}

void **printNotLoaded()** {

print out the no words were loaded warning;

}