# CS 31 Project 5

## Notable Obstacles:

* I was initially confused about how to run multiple files at once but eventually found that it was as simple as making an object file out of both program files
* My initial code to set minScore for each round weirdly returned 0 every time. I realized this was because I had set minScore to 0 at the start and every time it was compared to the new score it would be less than that (0<positive integer). To fix this, I added the condition that if minScore==0, have minScore take on the other value.
* When testing my program I created my own C-strings. However, my program returned nonsensical output (specifically it should have returned numbers like 0 or 1 but instead returned numbers in the 200s like 246 and 247). I realized this was because I had not included the null terminator in my C-strings which led to memory corruption
* Was initially using cin everywhere but later realized that I needed cin.getline in case the user typed in a space in their guess input

## Pseudocode for my stars.cpp program

#include relevant libraries for setprecision, input/output, cstring functions, and to use islower()function

void starsandplanets(secretWord c-string, user guessed word c-string){

Initialize counter variables to keep track of stars and planets

Initialize copy of secretWord to manipulate

Use strcpy() to copy secretWord onto copy variable

Initialize variables swSize and gwSize to store length of c-strings

for(all elements in the two c-strings up to the max index of the smaller cstring){

if(characters at the same index match up){

Change character in copy variable to a ‘\*’ to show that it has been counted

Increment starCount variable

}

}

for(all i less than size of guessWord){

if (i < secretWord’s size && guessWord[i] == secretWord[i]) {

Continue to next iteration;

}

for (all j less than size of secretWord){

if(guessWord[i]==copy[j]){ //remember that copy has some ‘\*’s in it to indicate already processed characters

Increment planet count;

Change copy[j] to a '\*' to show it has been analyzed;

Break; //prevents multiple identical characters from being changed to stars

}

}

}

Print stars and planets counts;

}

int runOneRound(const char words[][7], int nWords, int wordnum){

bool correctGuess=false; //flag

bool contains=false; //flag

char trialWord[100];

int trialWordSize =0;

int tries=0;

Check for basic errors like number of words in wordlist being less than or equal to 0, random integer being less than 0, or the random integer being greater than the number of words in wordList

while(user has not guessed right){

* Ask for trial word and cin.getline() input into trialWord variable
* If user input is less than 4 or greater than 6 characters return error message and ask for another guess
* Iterate through input. If there exist any non lower case characters (check with islower()), return error message and ask for another guess
* Check if user input word matches any word in word list
  + If not, say “I don’t know this word” and continue to next iteration of while loop
* Check if user input word matches up with secret word
  + If yes, increment tries and end function by returning number of tries
  + If no, run the starsandplanets function
* Increment tries and go on to next iteration

}

} //runOneRound function ends

int main(){

int rounds; //number of rounds

int score; //score for round

double totalScore; //total score that can be converted into average

int minScore; //minimum score

int maxScore; //maximum score

char secretWord[7]; //the secret word

int wordnum = 0; //random integer we will use use as index to get our secret word

Set precision of printed doubles to two decimal places

Initialize word list array of C-strings

Make variable to store wordfilename

Call getWords to populate wordlist with words from wordfile

If no words in wordList, end program and say no words were loaded

Ask user how many rounds they want to play

Check if number of rounds is positive

Use cin.ignore to erase any other input other than the integer

for(each round){

Choose a secret word from wordList using randInt

Print info (round number and number of letters in secret word)

Call runOneRound() function and add tries to score variable

If score is less than current minScore → change minScore

If score is more than current maxScore → change maxScore

Print number of tries it took user to guess word

Add score to totalScore and then calculate average score over rounds so far

Print average, minscore, maxscore, information

}

} //end main function