Project 1
<Hangman Game>

CIS-5
Tatiana DeJaco
02/02/15

Introduction

Title: Hangman Game

Hangman is a two player game where player one chooses a word at random and player two guesses its letters. Each time player two guesses incorrectly, player one draws a part of a stick figure. If player two guesses correctly, player one writes the letter where it is located in the word. The game continues until player two is able to reveal the word. However, if player one completes the drawing of a stick figure, then player two has lost the game.

I have altered the game so that instead of it being a two player game, it is now the computer versus a player. The computer chooses a word at random and the player guesses its letters. Also, I have given the user an option to ask for a hint. However, they can only have one hint. If they do ask for a hint, their score decreases by five points. Lastly, I've also changed how the player loses. The player loses by guessing incorrectly a total seven times. I chose to give them a strike limit of seven because in a stick figure there are a maximum of seven parts.

Summary

Project Size: 200+ lines
The number of variables: around 19
The number of method(s): 7

I tried to use all the concepts that were covered in the course but, in the end, I used: if statements, if-else statements, switch statements, arrays, for-loops, while-loops, and enumeration types, and overloaded functions.

The computer reads a word from a file containing a list of around 300 random words. At the end of the game, a file containing the chosen word and the user's results: whether they won or lost, the number of strikes, and their score.

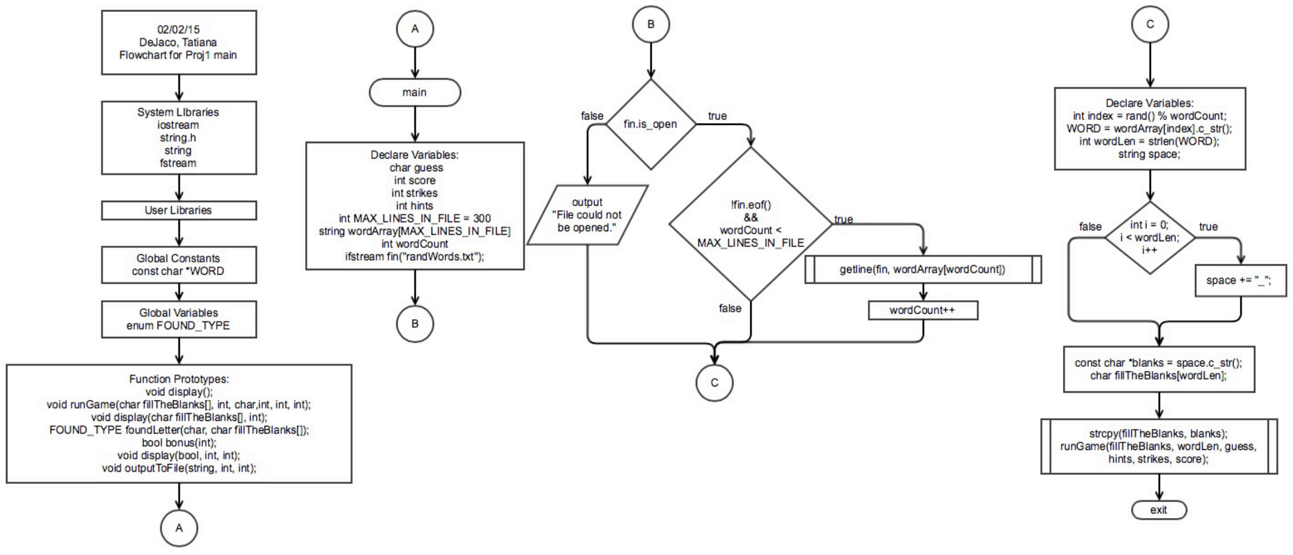
The project itself took about 5 days. I wanted to add an option for the user to switch the role of the computer. In other words, the option to chose the word and have the computer guess it's letters. Sadly, I ran out of time and was unable to implement it.

I used the libraries: iostream, string, and fstream.

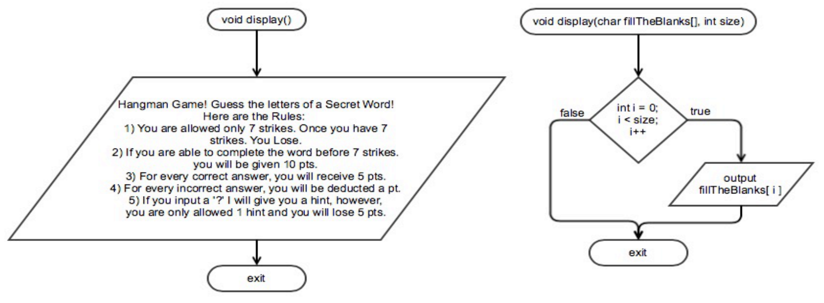
Pseudo Code

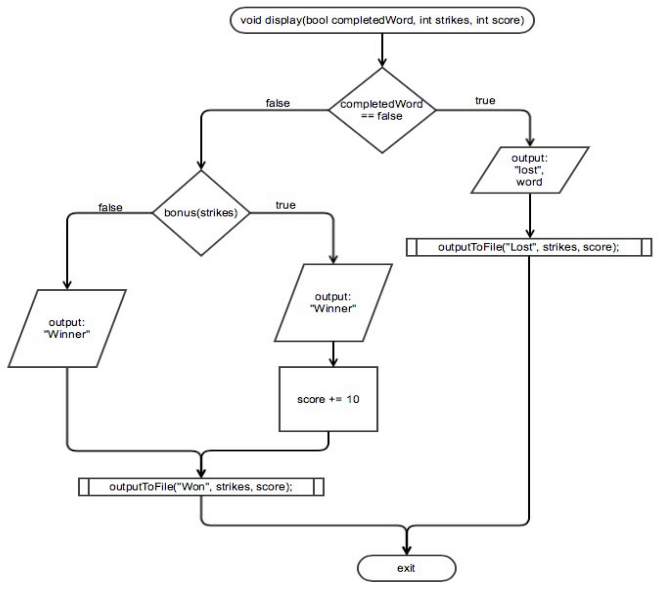
```
Initialize

read a file
choose a word from file

create the fill in the blanks

display the rules of the game
display the fill in the blanks

while the user has yet to completely fill in the blanks
    read: the user's guess
    if input is a ?
        if the number of hints used is equal to zero
            output: the number of vowels there are in the letter
            decrement their score by 5
        else
            output: no more hints
        increment the number of hints used
    else
        if the letter was not found
            output: incorrect! -1pt
            decrement their score
            increment their strike count
        else
            if the letter was found
                output: correct! +5 pts.
                Increment score by 5
            else
                output: letter was already found

            display the fill in the blanks
            if the word filled out is equal to the word chosen by the computer
                end the game user won
            else
                if strike count is equal to 7
                    end the game user lost
display the game results
```

FlowChart For Class Proj1 main(int argc, char** argv)

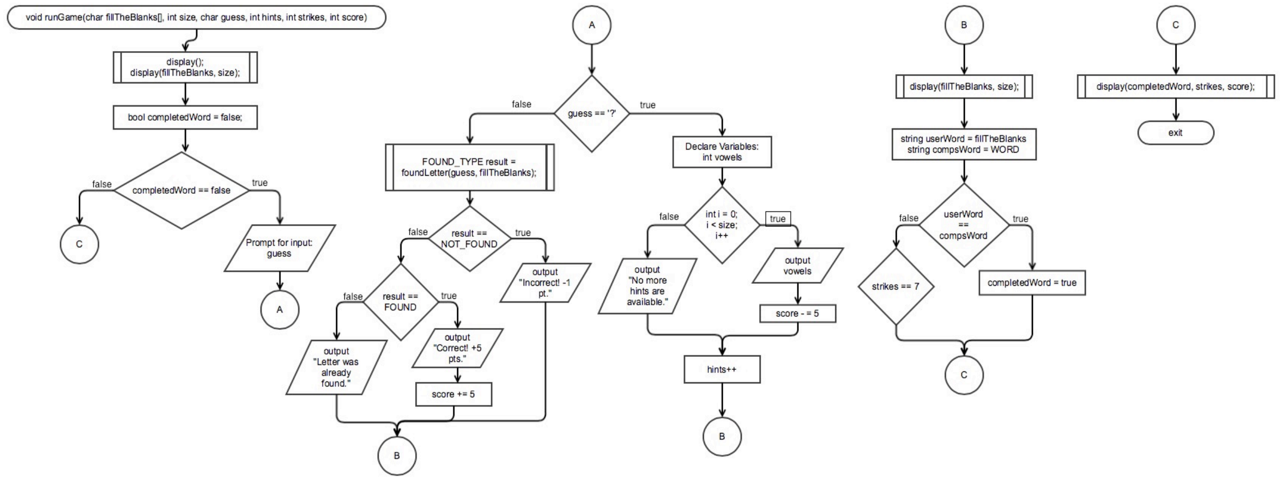

02/02/15
DeJaco, Tatiana
Flowchart for Proj1 main
System LIbraries
iostream
string.h
string
fstream
User Libraries
Global Constants
const char *WORD
Global Variables
enum FOUND_TYPE
Function Prototypes:
void display();
void runGame(char fillTheBlanks[], int, char,int, int, int);
void display(char fillTheBlanks[], int);
FOUND_TYPE foundLetter(char, char fillTheBlanks[]);
bool bonus(int);
void display(bool, int, int);
void outputToFile(string, int, int);
A
A
main
Declare Variables:
char guess
int score
int strikes
int hints
int MAX_LINES_IN_FILE = 300
string wordArray[MAX_LINES_IN_FILE]
int wordCount
ifstream fin("randWords.txt");
B
B
fin.is_open
false
true
output
"File could not
be opened."
!fin.eof()
&&
wordCount <
MAX_LINES_IN_FILE
true
false
getline(fin, wordArray[wordCount])
wordCount++
C
C
Declare Variables:
int index = rand() % wordCount;
WORD = wordArray[index].c_str();
int wordLen = strlen(WORD);
string space;
int i = 0;
i < wordLen;
i++
false
true
space += "_";
const char *blanks = space.c_str();
char fillTheBlanks[wordLen];
strcpy(fillTheBlanks, blanks);
runGame(fillTheBlanks, wordLen, guess,
hints, strikes, score);
exit

FlowChart For Class Proj1 void display(), void display(char fillTheBlanks[], int), void display(bool, int, int)

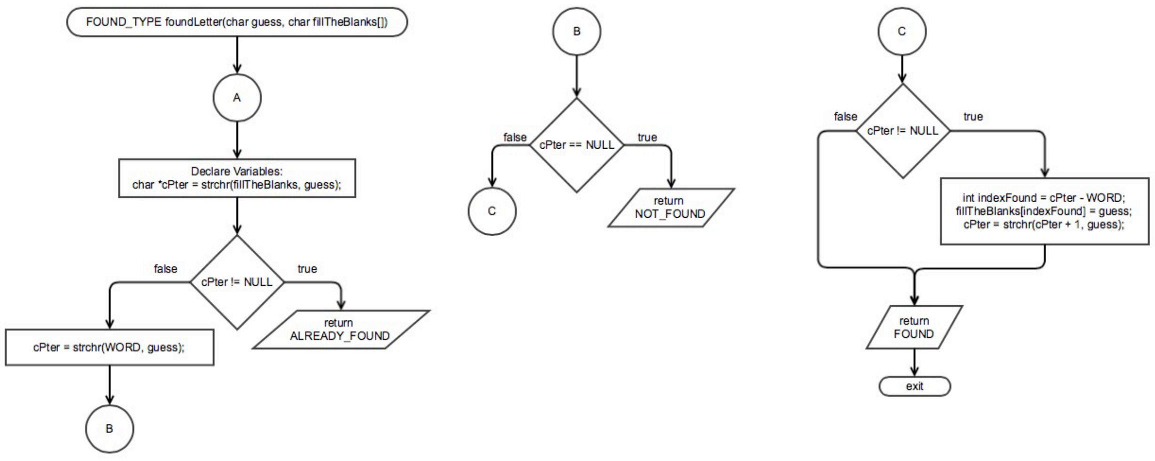

void display()
Hangman Game! Guess the letters of a Secret Word!
Here are the Rules:
1) You are allowed only 7 strikes. Once you have 7
strikes. You Lose.
2) If you are able to complete the word before 7 strikes.
you will be given 10 pts.
3) For every correct answer, you will receive 5 pts.
4) For every incorrect answer, you will be deducted a pt.
5) If you input a '?' I will give you a hint, however,
you are only allowed 1 hint and you will lose 5 pts.
exit
void display(char fillTheBlanks[], int size)
int i = 0;
i < size;
i++
false
true
output
fillTheBlanks[i]
exit


void display(bool completedWord, int strikes, int score)
completedWord
== false
false
true
output:
"lost",
word
outputToFile("Lost", strikes, score);
bonus(strikes)
false
true
output:
"Winner"
output:
"Winner"
score += 10
outputToFile("Won", strikes, score);
exit

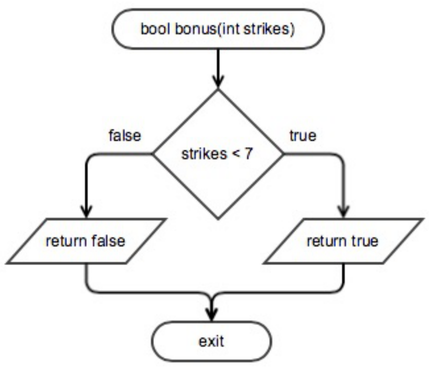
FlowChart For void runGame(char fillTheBlanks[], int, char,int, int, int)


void runGame(char fillTheBlanks[], int size, char guess, int hints, int strikes, int score)
display();
display(fillTheBlanks, size);
bool completedWord = false;
completedWord == false
false
true
C
Prompt for input: guess
A
A
guess == '?'
false
true
FOUND_TYPE result = foundLetter(guess, fillTheBlanks);
result == NOT_FOUND
false
true
result == FOUND
false
true
output "Letter was already found."
output "Correct! +5 pts."
score += 5
output "Incorrect! -1 pt."
B
Declare Variables: int vowels
int i = 0; i < size; i++
false
true
output "No more hints are available."
output vowels
score - = 5
hints++
B
B
display(fillTheBlanks, size);
string userWord = fillTheBlanks
string compsWord = WORD
userWord == compsWord
false
true
strikes == 7
completedWord = true
C
C
display(completedWord, strikes, score);
exit

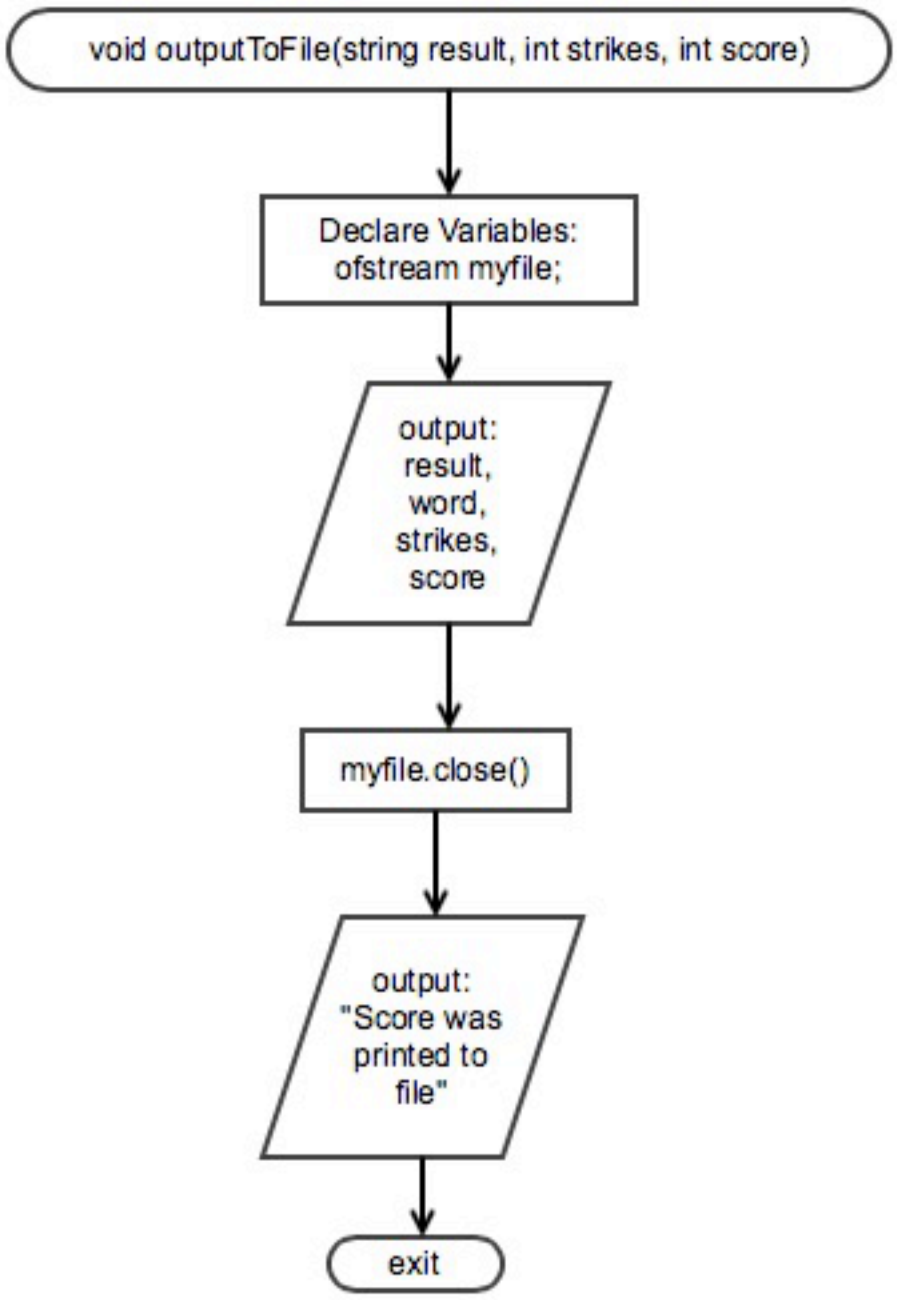
FlowChart For FOUND_TYPE foundLetter(char, char fillTheBlanks[])


FOUND_TYPE foundLetter(char guess, char fillTheBlanks[])
A
Declare Variables:
char *cPter = strchr(fillTheBlanks, guess);
cPter != NULL
false
true
cPter = strchr(WORD, guess);
return ALREADY_FOUND
B
B
cPter == NULL
false
true
C
return NOT_FOUND
C
cPter != NULL
false
true
int indexFound = cPter - WORD;
fillTheBlanks[indexFound] = guess;
cPter = strchr(cPter + 1, guess);
return FOUND
exit

FlowChart For bool bonus(int)



FlowChart For outputToFile(string, int, int)


void outputToFile(string result, int strikes, int score)
Declare Variables: ofstream myfile;
output: result, word, strikes, score
myfile.close()
output: "Score was printed to file"
exit

Major Variables

Type	Variable Name	Description	Location
const char	WORD	The string that the computer chose at random	Class Proj1
char	guess	The user's input	main(int argc, char** argv)
int	score	The user's current score	main(int argc, char** argv)
	strike	How many times the user guessed incorrectly	main(int argc, char** argv)
	hints	How many times the user asked for a hint	main(int argc, char** argv)
	wordLen	The length of the word chosen by the computer	main(int argc, char** argv)
const char[]	fillTheBlanks	The user fills in the blanks as they guess correctly	main(int argc, char** argv)

C++ Constructs

Chapter	New syntax and Keywords	Location
2	cout	runGame(char, int, char, int, int, int);
	cin	runGame(char, int, char, int, int, int);
	int	main(int argc, char** argv)
	char	main(int argc, char** argv)
	bool	runGame(char, int, char, int, int, int);
	string	main(int argc, char** argv)
	assignment operator (+=)	runGame(char, int, char, int,

		int, int);
	arithmetic operator (+,-,*, /)	runGame(char, int, char, int, int, int);
	increment operator (++)	runGame(char, int, char, int, int, int);
	decrement operator (--)	runGame(char, int, char, int, int, int);
3	enumeration type	Class Proj1
	if-else statements	bonus(int)
	switch statements	runGame(char, int, char, int, int, int);
	break	runGame(char, int, char, int, int, int);
	while loops	runGame(char, int, char, int, int, int);
	for loops	runGame(char, int, char, int, int, int);
4	functions	Class Proj1
	global constant	Class Proj1
	global variable	Class Proj1
	overloaded functions	runGame(char, int, char, int, int, int);
5	void functions	Class Proj1
	return	bonus(int)
	procedural abstraction	runGame(char, int, char, int, int, int);
6	ifstream	outputToFile(string, int, int)
	ofstream	outputToFile(string, int, int)
7	arrays	main(int argc, char** argv)

Reference

1. http://www.stackoverflow.com
2. http://www.cplusplus.com

3. textbook

Program

```
/*
 * File:   main.cpp
 * Author: Tati
 * Created on January 28, 2015, 10:12 PM
 * Hangman Game
 */

#include <iostream>
#include <string.h>
#include <string>
#include <fstream>
using namespace std;

const char *WORD; // The string the computer has chosen
// Used for comparing results
enum FOUND_TYPE { NOT_FOUND, FOUND, ALREADY_FOUND };

// Displays the rules of the game
void display();
// Run the game
// Pass in the blanks to be filled, it's size, guess, hints, strikes, & score
void runGame(char fillTheBlanks[], int, char,int, int, int);
// Display the fill in the blanks
void display(char fillTheBlanks[], int);
// Look for the letter given by user
// Pass in guess, the blanks to be filled
FOUND_TYPE foundLetter(char, char fillTheBlanks[]);
// See if user completed word in less than 7 strikes
bool bonus(int); // Pass in their number of strikes
// Output if the user won or lost
// Pass in if word was completed, # of strikes, & score
void display(bool, int, int);
// Output the word, # of strikes, and score to a file
// Pass in their result, their # of strikes, and ttl score
void outputToFile(string, int, int);

int main(int argc, char** argv)
{
   // initialize random seed
   srand (time(NULL));

   // Declare Variables
   char guess;     // the user's guess
   int  score   = 0; // the user's score
   int  strikes = 0; // how many times they guessed wrong
   int  hints   = 0; // how many hints the user used up

   const int MAX_LINES_IN_FILE = 300;
   string wordArray[MAX_LINES_IN_FILE]; // allocating an array of 1kb
   int wordCount = 0; // counts the number of word in file
   // Computer will choose one word from the file of random words
   ifstream fin("randWords.txt"); // opening an input stream for file
   // Checking whether file could be opened or not. If file does not exist
   // or don't have read permissions, file stream could not be opened.
   if(fin.is_open())
   {
      // this loop run until end of file (eof) does not occur
      while(!fin.eof() && wordCount < MAX_LINES_IN_FILE)
      {
         // Read a complete line into the array. Each line
         // contains one word.
         getline(fin, wordArray[wordCount]);
         wordCount++;
      }
   }
   else //file could not be opened
      cout << "File could not be opened." << endl;

   // Display a random word from file
   // Pick a string from the list
   int index = rand() % wordCount;
   WORD = wordArray[index].c_str(); // Set the rand string to be the word
   int wordLen = strlen(WORD);      // the length of the chosen string

   // Create empty string that will be filled as the user guesses correctly
   string space;
   for (int i = 0; i < wordLen; i++)
      space += "_";
   const char *blanks = space.c_str(); // Create the empty string
   char fillTheBlanks[wordLen]; // The user will fill as they guess correctly
   strcpy(fillTheBlanks, blanks);

   runGame(fillTheBlanks, wordLen, guess, hints, strikes, score);
   return 0;
}

// Displays the rules of the game
void display()
{
   cout << "Hangman Game! Guess the letters of a Secret Word!"          << endl;
   cout << "Here are the Rules: "                                       << endl;
   cout << "1) You are allowed only 7 strikes. Once you have 7"        << endl;
   cout << "   strikes. You Lose."                              << endl;
   cout << "2) If you are able to complete the word before 7 strikes." << endl;
   cout << "   you will be given 10 pts."                        << endl;
   cout << "3) For every correct answer, you will receive 5 pts."      << endl;
   cout << "4) For every incorrect answer, you will be deducted a "    << endl;
   cout << "   pt."                                         << endl;
   cout << "5) If you input a '?' I will give you a hint, however,"    << endl;
   cout << "   you are only allowed 1 hint and you will lose 5 pts."   << endl;
   cout << endl;
}

// Run the game
void runGame(char fillTheBlanks[], int size, char guess, int hints, int strikes, int score)
{
   // Display the rules of the game
   display();
   // Display how many letters are in the word
   display(fillTheBlanks, size);

   bool completedWord = false; // true if user completely filled the blanks
   while (completedWord == false)
   {
      // Prompt user for input
      cout << "Your guess? ";
      cin >> guess;

      // If user asks for a hint, give them one at random
      if (guess == '?')
      {
         if (hints == 0)
         {
            // Tell them how many vowels are in the word
            int vowels;
            // Go through the WORD and count how many vowels are in there
            for (int i = 0; i < size; i++)
            {
               switch(WORD[i])
               {
                  case 'a': { vowels++; break; }
                  case 'i': { vowels++; break; }
                  case 'u': { vowels++; break; }
                  case 'e': { vowels++; break; }
                  case 'o': { vowels++; break; }
                  default: break;
               };
            }
            cout << vowels << " vowel(s) is in this word." << endl;
            cout << "-5 pts" << endl;
            score -=5;
         }
         else
            cout << "No more hints are available." << endl;

         hints++;
      }
      else // if user didn't ask for a hint
      {
         // Find the letter guessed by user in the word
         FOUND_TYPE result = foundLetter(guess, fillTheBlanks);
         if (result == NOT_FOUND)
         {
            cout << "Incorrect! -1 pt.";
            score--;
            strikes++;
         }
         else
         {
            if (result == FOUND)
            {
               cout << "Correct! +5 pts.";
               score += 5;
            }
            else
               cout << "Letter was already found.";
         }
         cout << endl << endl;

         // Show user where their guess was filled in
         display(fillTheBlanks, size);

         // Check if the blanks are filled and matches the computer's word
         string usersWord = fillTheBlanks; // User's word
         string compsWord = WORD;          // Comp's word
         if (usersWord == compsWord) // if blanks match the comp's word
            completedWord = true;
         else // blanks were not filled
         {
            if (strikes == 7) // Check if the user used up their strikes
               break;
         }
      }
   }
   // Display results to user
   display(completedWord, strikes, score);
}

// Display fill in the blanks
void display(char fillTheBlanks[], int size)
```

```
{
   // Display the blanks
   for (int i = 0; i < size; i++)
      cout << " " << fillTheBlanks[i];
   cout << endl;
}

// Look for the letter given by user
FOUND_TYPE foundLetter(char guess, char fillTheBlanks[])
{
   // If the guess letter is in the string fillTheBlanks
   // then the user already guessed that character.
   char *cPter = strchr(fillTheBlanks, guess); // char pointer
   if (cPter != NULL)
      return ALREADY_FOUND;

   // If it is NOT in the fillTheBlanks,
   // then check if character is in the word
   cPter = strchr(WORD, guess);
   if (cPter == NULL)
      return NOT_FOUND;

   // If character WAS not in the fillTheBlanks
   // and character WAS found, then it is in the word.
   // So...get it's location in the word.
   while (cPter != NULL)
   {
      // FIll in the blank with the guess
      int indexFound = cPter - WORD;
      fillTheBlanks[indexFound] = guess;
      // Advance to the next location of guess WORD
      cPter = strchr(cPter + 1, guess);
   }
   return FOUND;
}

// See if user completed word in less than 7 strikes
bool bonus(int strikes)
{
   if (strikes < 7)
      return true;
   else
      return false;
}

// Display results of game and output results to a file
void display(bool completedWord, int strikes, int score)
{
   cout << endl;
```

```
    // Set Score and Output results
    if (completedWord == false) // Word wasn't completed
    {
        cout << "You Lose!";
        cout << " The word was " << WORD;
        outputToFile("Lost", strikes, score);
    }
    else // Word was completed
    {
        if (bonus(strikes)) // user finished before 7 strikes
        {
            cout << "Amazing! You completed the word before 7 strikes! ";
            cout << "+10 pts!";
            score += 10;
        }
        else
            cout << "You have completed the game! Congratulations!";
        outputToFile("Won", strikes, score);
    }
}

// Output the word, # of strikes, and score to a file
void outputToFile(string result, int strikes, int score)
{
    // Output a file when game is complete
    ofstream myfile;
    myfile.open("hangman.docx");
    myfile << "You " << result << " the game!"  << endl;
    myfile << "The word was: " << WORD          << endl;
    myfile << "You used up "   << strikes       << " strikes" << endl;
    myfile << "Your score = "  << score         << endl;
    myfile.close();
    // Tell User their result was outputted to a file
    cout << endl;
    cout << "Your score was printed to a file";
    cout << "...Go check out your score!" << endl;
}
```