Assignment 02

Due: Friday, Feb. 11, 2021, at midnight, 100 points.

Documentation Header Reminder

Before you start your assignment, you will need to add documentation similar to what we demonstrated in the first few lectures.

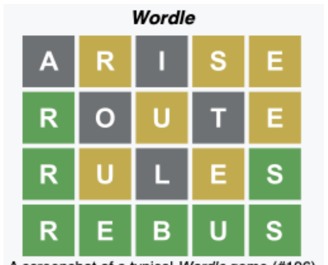
```
// Programmer: Clayton Price
// Date: 9/4/1002.
// File: fahr2celc.cpp
// Assignment: HW2
// Purpose: this file contains the main function of the program which
//    will input Fahrenheit temps from the user, then convert and
//    output the corresponding Celcius temperature.
```

Word e For Frie d

Background

Have you heard of "Wordle"? If you are the type of person who enjoys their daily dose of news, then you must have (or if you use Twitter a lot). Wordle is a rather simple web-based word game, and players love to share their success, or failure, in the game with their friends. So where does the name of the game come from? You guessed it, the game is developed by Welsh-born software engineer Josh Wardle. And you know what? The game was purchased by The New York Times Company in January 2022 for a seven-figure sum! Wait…so why am I still wasting time writing this assignment? I should be coming up with new games to sell! I guess I'll do that after writing this up…

So to play the original game, players have six attempts to guess a five-letter word; feedback is given for each guess, in the form of colored tiles, indicating when letters match or occupy the correct position. Here is a sample image for how the actual game looks like:



A screenshot of a typical *Wordle* game (#196)

Looks fun, right? In this assignment, we will be mimicking the Wordle game with a simpler version, using the C++ techniques and tools that we have learned so far.

Specifications

Your C++ program (all code can still be contained within the main function) should be designed to be played by two players instead of one. The main objective is still to have one of the players guess a five-letter word correctly. The program should first prompt Player 1 to enter "the word" for Player 2 to guess, after asking for their names of course. At the start of the program, it should also ask Player 1 to enter the *friendship level*, a higher value means a closer friendship bond with Player 2. Friendship level should have a range of value between 0 and 100, inclusive. If Player 1 inputs something outside of the acceptable range, then the program should keep prompting the player until a proper value is entered.

Just like the original Wordle game, Player 2 will have a maximum of **six** attempts to guess the word correctly. In each attempt, the program should prompt Player 2 to enter their word, and then see if the entered word matches with the key. If it's an exact match, then the program should output a congratulations message so the two players can move onto their next friendship bonding activity. However, if it's not a perfect match, then the program should provide some useful feedback as in the original game using the following notations:

- Notation **`@.@`** should be used if an input letter is not a part of the key
- Notation **`^( '-' )^`** should be used if an input letter is a part of the key but is not in the right position
- Notation **`^o^`** should be used if an input letter is a part of the key and is at the right position

For example, if the key word is "beard" and Player 2 entered "bound" as the first attempt, then the feedback provided should be:

```
You entered "BOUND". Nice try, keep going!
```

```
B: ^o^
O: @.@
U: @.@
N: @.@
D: ^o^
```

The completion of all six attempts concludes a round. If Player 2 guessed the key word correctly, then the friendship level value should be correspondingly *increased* by the number of attempts Player 2 took to get to the correct word. Note that the maximum value friendship level can reach is 100. Next, the program should ask Player 1 if they want to begin a new round. If no, then the program will then keep asking Player 1 if they want to start a new round with a different Player 2. If Player 1 inputs no again, then the program will exit smoothly.

Additional Info

- This program should trust no one. When prompting Player 1 for the key word, we need to check a couple things.
    - First, we need to ensure that the key word is a valid five-letter word (you may find this string function to be useful). If this constraint is violated, then the program should keep prompting Player 1 for the proper key word until one is entered.
    - Besides checking for the proper length of the key word, we also need to make sure that the key word is valid, i.e. each letter of the word is an alphabet. How do you do that? Well maybe this function can help you (depending on your C++ compiler, you may need to include the <ctype.h> library).
        - But what if Player 1 input a word that is not an actual word that exists in the dictionary? Well, that's the best we can do at this point, and checking for the meaning is outside of the scope of this assignment. **Ten bonus points** if you are able to come up with a good coding solution to address this issue. .
- For this assignment, you may also find it to be useful to know how to manipulate (i.e. read/write) each letter in a string variable. One easy way to do that is through the use of the "at" function for the string data type. A sample usage is provided here:

```
#include <string>
#include <iostream>
using namespace std;
int main(){
     string str = "LAUGH";
     // note: the first character in a string is
     //   located at index position 0 (not 1)
     for(int index=0; index<str.length(); index++){
          cout << str.at(index) << endl;
     }
}
```

- To make this assignment a bit easier for when you are checking if two words are an exact match, you can expect all of the input words from the players to be in uppercase. That is, you don't have to perform input validation to check if the words contain any lowercase letters.

Sample Output

Coming soon...