CS 10 - Assignment 7: Hangman

Collaboration Policy

We encourage collaboration on various activities such as lab, codelab, and textbook exercises. However, **no collaboration between students is allowed on the programming assignments**. Please be sure to read and understand our full policy at: Full Collaboration Policy

Submission Instructions

Submit to R'Sub testing, feedback and grading.

Assignment Specifications

The game called Hangman is a spelling game: one player sets up a word or phrase, represented by a row of dashes. If the guessing player suggests a letter which occurs in the word, the other player writes it in all its correct positions. If the suggested letter does not occur in the word, the other player draws one element of the hanged man stick figure as a tally mark. The game is over when:

- The guessing player completes the word, or guesses the whole word correctly
- The other player completes the diagram.

You must implement a simplified version of hangman in which the player gets **7 incorrect guesses** (corresponding to drawing the "hanged man").
You are required to implement the specified functions and utilize any provided functions.

Hangman Algorithm

- Get phrase to be solved from the user (using `getline` instead of the input operator `>>` to allow for the presence of spaces in the phrase)
- Clear output window so phrase is not seen, using the provided function `clearScreen()`
- Setup <u>unsolved phrase</u> so game has a separate string with dashes in place of characters (call `setupUnsolved` function)
- Display the unsolved phrase
- Play Game until phrase is completely guessed, or all 7 incorrect guesses are used up
    - Get a valid letter guess from user (call `getGuess` function - see below for meaning of "valid guess")
      (After initial phrase has been set, user input should **ONLY** occur in the `getGuess` function)
    - Update a string consisting of *<u>all characters guessed so far</u>*
    - Take action on the guess:
        - Correct: update unsolved phrase using the `updateUnsolved` function
        - Incorrect: increment the wrong guess count
    - Clear the screen again, using `clearScreen()`
    - Output game status:
        - updated unsolved phrase - phrase with dashes for still-unguessed letters
        - list of characters guessed so far
        - number of wrong guesses left
- Output game over message (`You lost!` or `Congratulations!!`)

Header File

We have supplied a header file called `assn.h` to be utilized with this assignment: it provides the function `clearScreen()` that will clear the terminal window, to be invoked at specified spots within the algorithm. You should acquire this file, either run `git pull` on cloud9 if you cloned the cs010_assignments workspace or upload this file into your Hangman assignment directory, and add it to your includes:

```
#include "assn.h"
```

(Note the quotes, unlike the <> used for including C++ libraries)

Additional Requirements and Hints

- **Implement each function** and get feedback/credit for each **before doing main game**
- **All** user input statements should be followed by outputting a blank line.
- **All user inputs and corresponding prompts** after initial phrase setup **occur only in body of** `getGuess` - so there are no input or output statements in `setupUnsolved` or `updateUnsolved`
- All phrases and valid guesses will use lowercase alphabetic characters *(i.e. in our testing, we will only use lower case phrases for the initial puzzle; and we will enter only lower case characters as **valid** guesses; we will, of course, input non-alpha characters to test for **invalid** guesses).*
- The player gets **7** wrong guesses
- You should be utilizing a single `puzzle` variable to hold the current state of the game, and assigning to it the return value from `setupUnsolved` initially, and `updateUnsolved` subsequently.
- **Use the string member function `at()`**: Remember, `at()` can be used on either side of the assignment operator (=), to either "read" the value at a specified position of the string (right side); or to "write" a new value into a position of the string (left side).
  The function `at()` "throws an exception" when you attempt to access a non-existent location, unlike the `[]` syntax; this can be a great help when debugging.
- **`bool isalpha(char)`**: In order to check if a guess is valid - i.e. if it is an alphabetic character - you can use this built-in predicate function that takes a single character as an argument. Requires that you include the `<cctype>` library.
- ***Note:*** The functions `setupUnsolved` and `updateUnsolved` are "complementary" to each other (one replaces letters with dashes, the other replaces those dashes with letters).
  So once you've got one working, the other should be real simple!

Functions

The following 3 functions are **required, exactly as declared**.
You may, if you wish, define other functions in addition to these.

You must define, implement and correctly invoke these 3 functions exactly as declared here. You may wish to copy & paste the following into your source code, and comment out the function declarations you are not yet ready to implement.

```
/// @brief Puts dashes in place of alphabetic characters in the phrase.
/// @param phrase the phrase to be solved
/// @return the phrase with all alphabetic characters converted to dashes
string setupUnsolved(string phrase);


/// @brief Replaces the dashes with the guessed character.
/// @param phrase the phrase to be solved
/// @param unsolved the phrase with dashes for all unsolved characters
/// @param guess the char containing the current guessed character
/// @return the new unsolved string with dashes replaced by new guess
string updateUnsolved(string phrase, string unsolved, char guess);


/// @brief Gets valid guess as input.
///
///    A guess is taken as input as a character. It is valid if
///    1) it is an alphabetic character; and
///    2) the character has not already been guessed
///
/// @param prevGuesses the string containing all characters guessed so far
/// @return a valid guess and only a valid guess as a character
char getGuess(string prevGuesses);
```

Reading `diff` Output

Rather than list the complete outputs of your program and the solution, we show you only the differences between them by running a program called diff.
Here is a sample diff message, with a guide to reading it:

```
--- hangman.cpp
+++ solution.cpp
@@ -56,2 +56,2 @@
 wrong guesses left: 0
-you lost
+you lost!
```

The **first 2 lines are a key**, showing the symbol associated with each file;
the line enclosed in "`@@`" symbols describes the following difference list, which details one difference between the program outputs.
The **first character in a difference line indicates which file it comes from**. If there is no leading symbol ('+' or '-'), then the line is present in both program outputs.

So the diff output in the example means:

The `-` symbol is associated with the file named `hangman.cpp`
The + symbol is associated with the file named `solution.cpp`
The line "`@@ -56,2 +56,2 @@`" means: following are **2** lines from hangman.cpp ('-') starting from line **56**; and **2** lines from solution.cpp ('+'), also starting from line **56**.
The line "`wrong guesses left: 0`" has **no leading symbol, it occurs in both files**.
The line "`-you lost`" has a **leading minus sign - it occurs in hangman.cpp.**
The line "`+you lost!`" has a **leading plus sign - it occurs in solution.cpp.**

To fix the above program, we note that line 57 of hangman.cpp is missing an exclamation mark; if we add it then the output will match, and this diff message will disappear.

Example Run

(user input has been **bolded and underlined** for emphasis; c by itself is `clearScreen output`.
Note the two invalid guesses: ‘l’ is invalid because it is already in the list of previously guessed characters; and 8 is invalid because it is not alphabetic
Note also that the original non-alphabetic character ‘.’ was not replaced by a dash)

```
Enter phrase: hello.

c

Phrase: -----.

Enter a guess: h

c

Phrase: h----.

Guessed so far: h
Wrong guesses left: 7

Enter a guess: e

c

Phrase: he---.

Guessed so far: he
Wrong guesses left: 7

Enter a guess: l

c

Phrase: hell-.

Guessed so far: hel
Wrong guesses left: 7

Enter a guess: d

c

Phrase: hell-.

Guessed so far: held
Wrong guesses left: 6

Enter a guess: l

Invalid guess! Please re-enter a guess: 8

Invalid guess! Please re-enter a guess: o

c
```

```
Phrase: hello.

Guessed so far: heldo
Wrong guesses left: 6

Congratulations!!
```