1 Assignment 6 Introduction

This assignment is due *before* 11:59pm on the listed date. Which means, submissions made *on or after* 11:59pm will be counted as a late submissions. Late submissions *before* 11:59pm on the following day will receive a 25% point deduction as penalty. Submissions made *on or after* 11:59pm on the same day will *not* be graded. We strongly recommend completing the assignment before then.

It is imperative that you meticulously follow the submission process outlined at the end of this assignment. Incorrectly structured submissions will receive a 10% point deduction as a penalty.

Assignments due on Mondays generally involve materials covered in lectures from the previous week, whereas assignments due on Thursdays involve materials covered in lectures from the running week. So be sure to watch the lectures and go over the reading materials before attempting the assignments.

Good luck!

2 Shared Letters (50 points)

Write a function named `SharedLetters` that takes in three strings (the first by reference, the second by copy, and the last by a pointer) as function parameters, and returns a string. The function should compare the letters in the string by position, and for each position, report the number of pairs of strings that share a letter, separated by a comma. Note that this function should not change any of its arguments, so all the function parameters should be const.

Your `main` function should only read in three words from standard input, call the `SharedLetters` function with the appropriate parameters, and output the result. Your `SharedLetters` function declaration (along with any other functions declarations) should be in a header file named `share.h`, with their corresponding definitions in the appropriate implementation file `share.cpp`.

Input:

```
meetings
resting
rectangle
```

Output:

```
1,3,0,3,1,3,3,0,0,
```

In the above example, the first value in the result is `1`, because `resting` and `rectangle` share the letter `r` in the first position. The next value is `3`, because all three possible pairs share the letter `e` at the second position (`resting-meetings`, `meetings-rectangle`, and `resting-rectangle`). The next value is `0`, because no pairs match at the third position. Note that the words may not be of the same length, so be mindful not to index beyond the end, and use the `.at()` method, instead of `[]` to index.

Some more test cases are provided below:

Input:

```
m
s
u
```

Output:

```
0,
```

Input:

```
sparty
moriarty
apart
```

Output:

```
0,1,1,1,1,0,0,0,
```

Input:

```
beef
bee
be
```

Output:

```
3,3,1,0,
```

General Instructions:

- When writing your program, you may only use concepts you have learned in the course thus far.
- Your program should compile and run. You can assume there will be exactly three lower-case words (one in each line) of at least length of 1.
- The fucntion `SharedLetters` must take in the function parameters exactly as described in the problem statement, in the exact order.
- The header files and the implementation files must be created as specified in the problem statement.

3 Wordle (50 points)

Write a program that plays a simple game of Wordle. The first word provided is the secret 5 letter word that the player is trying to guess. After each guess, the program displays a string as response, which consists of the letters that were in the correct position. If the letter was present in the word but not in the right place, replace that letter with a `'?'`, else put a `'.'` if the letter is not present in the secret word at all. The player has 6 guesses before the game ends. Display the win or loss condition at the end of the game. The game can only end early if the player provides the correct word.

Your program should use a function `WordleResponse` that takes in a secret word and a guess as the function parameters, and returns a string as response. Note that this function should not change any of its arguments, so all the function parameters should be `const`.

Your `main` function should only read the words from standard input, call the `WordleResponse` function with the appropriate parameters, and either continue the game or display the win or lose condition. Your `WordleResponse` function declaration (along with any other functions declarations) should be in a header file named `wordle.h`, with their corresponding definitions in the appropriate implementation file `wordle.cpp`.

Some example inputs are provided below, following by usage:

Example Input:

```
HOUSE
MOUSE
GUESS
TRIAL
DOUBT
CRAFT
HOTTE
```

Example Usage:

```
Give me a secret word: HOUSE
Give me a guess: MOUSE
.OUSE
Give me a guess: GUESS
.??S?
Give me a guess: TRIAL
.....
Give me a guess: DOUBT
.OU..
Give me a guess: CRAFT
.....
Give me a guess: HOTTE
HO..E
You lose!
```

In the example above, `HOUSE` is the secret word. The first guess is `MOUSE`. Because `HOUSE` does not have an `'M'`, it is replaced with a `'.'`. The rest of the letters match. When the guess is `GUESS`, The `'U'` and the `'E'` are in the wrong position, so they become `'?'`. The first `'S'` is in the right position, and is shown, but the second `'S'` is not in the right position, but it is present in the secret, so it becomes a `'?'`. Since the correct word was not guessed within 6 tries, 'You lose!' was displayed, before terminating the program. If the correct word was guessed within 6 tries, 'You win!' would have been displayed instead, before terminating the program.

General Instructions:

- You can assume all words provided will be in upper-case and 5 letters long.
- When writing your program, you may only use concepts you have learned in the course thus far.
- Your program should compile and run.
- The fucntion `WordleResponse` must take in the function parameters exactly as described in the problem statement, in the exact order.
- The header files and the implementation files must be created as specified in the problem statement.

4 Assignment 6 Submission Process

- Create a folder, name it `your_msu_id6`. For example, if your MSU email is `johndoe@msu.edu`, then you should name the folder `johndoe6`.
- For each programming task, create a sub-folder inside your `your_msu_id6` folder, and name it as the number that corresponds to the programming task number. For this assignment, there should be two sub-folders named '2' and '3'.
- Inside each sub-folder, put the implementation files (along with any necessary header files) for the appropriate solution.
- Compress/Zip `your_msu_id6` folder and name it `your_msu_id6.zip`. For example, if the name of your folder is `johndoe6`, then you need to create a zip file named `johndoe6.zip`. Zip file guide: `https://copyrightservice.co.uk/reg/creating-zip-files`.
- Submit the zip file through D2L.