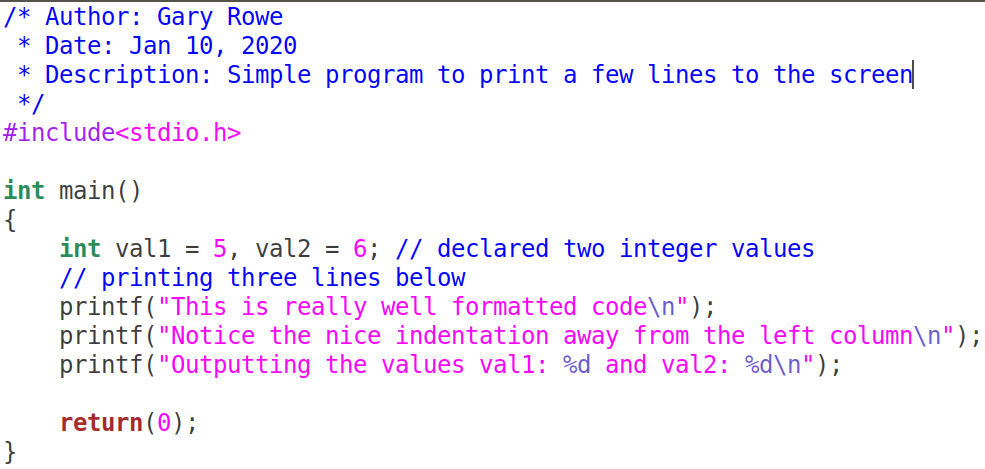
**Student Name:** **Weight: .**

**Student ID:** **Marks: \_\_\_\_\_\_\_\_\_\_**

**CREATE AN EMPTY DOCUMENT TO SUBMIT YOUR SOLUTIONS. DO NOT USE THIS DOCUMENT TO SUBMIT YOUR ANSWERS. YOU WILL LOSE 10% FOR DOING SO!!!**

Take Home Quiz1:

Your C files MUST be properly ***formatted*** with ***indentations*** that enhances your code ***readability***. Example of properly formatted code:

## Problem 1:

### Write a program that mimics features of a dictionary. The program will meet the following criteria:

1. Use the following array as your baseline.
   1. You should make your array capable of holding a maximum of 100 words.
   2. You should keep track of how many words are in your dictionary.
   3. Run the program until 100 words are in the dictionary or the user enters the number “2021”.

**arrChar[][20]={{"Around"}, {"Pool"}, {"LifeGuard"}, {"Strangers"}, {"Ordered"}, {"Marbles"}, {"Somewhere"}, {"Futuristic"}, {"Unrealistic"}, {"Definition"}, {"Surprise"}};**

1. Add **appropriate** words to enhance the program.
2. The main function should not perform any operation other than calling other functions.
   1. Your program should ask the user to enter a string, maximum 19 characters.
   2. The program will print the row that the string was found on.
3. You will create a function to get the user’s input:

**Required Function prototype:**

***void getInput(char \*input);***

1. The program should then call the function called **checkDictionary**.

**Required Function prototype:**

***int checkDictionary(char arrChar[][20], char \*guess, int rows);***

* 1. rows should convey how many rows are in your dictionary
  2. arrChar[ ][20] should be a reference to your dictionary, when passed by reference from main
  3. \*guess is a reference to the user’s string entered.
  4. The return value will be the row that the string entered was located.
  5. The function should perform this search insensitive to case. In other words **String**, **string**, **STRING** will all be treated as the same.

1. Each time you run the program, ask the user if they would like to enter more words.
   1. Using the rows variable as a way to track how many words are in the dictionary.
   2. You can then insert more words into the arrChar; if rows is less than 100.

**Required Function prototype:**

***void InsertWord(char arrChar[][20], int rows);***

1. You can create your own functions to change the case of a word or you can use a library function that performs string comparison with the case insensitive feature.

**Submission: Submit the completed code and screenshots of the working program**

## Problem 2:

### Download the C file provided on D2L. The program makes use of pointers to call a few functions.

The following is a list of all the functions, their description and the functionality they provide. Some functions are already complete and do not need to be altered. Other functions will require some modification to perform their intended task.

1. **swap**:
   1. **Purpose**: Takes the address of two strings and if the character pointed to by left pointer has a higher value that of the right they must be swapped. The function currently only analyses the first character.
   2. **Required**: This function is incomplete as it does not perform the task for every character in the string. You will fix the function so that it performs this swap for all characters in the string. You must stop at the NULL that come first, meaning the shorter string will determine when to stop swapping.
2. **reverse**:
   1. **Purpose**: Takes the characters in the string and reverses the entire string.
   2. **Required**: This function is **complete** and **does not require any modifications**.
3. **title\_rev**:
   1. **Purpose**: Takes an address to a string and reverses it, then converts the string to a title cased string
   2. **Required**: This has not been implemented and requires you to build it from scratch
4. **clr\_input:** 
   1. **Purpose**: Takes a string and replaces the '\n' character, if found, with the NULL terminator.
   2. **Required**: This is a helper function that does not need to be modified.
5. **getInput**:
   1. **Purpose**: Gets the user input string(s)
   2. **Required**: Currently the function can only retrieve 1 string at a time. Modify the function so that it takes 2 strings from the user. This means the function should be called only once.
6. **main**:
   1. **Purpose**: This is where you will call your other functions. Modify it as necessary without adding any major logic to it.
   2. **Required**:Call other functions, possibly declare other variables.

**Submission: Submit the completed code and screenshots of the working program**