CS 10 - Assignment 2

Collaboration Policy

Collaboration between students on programming assignments is **NOT** allowed under any circumstances - you may not even discuss your work with others; you may **NOT** get coding assistance, or copy code, from **ANY** outside source *(books, web sites, current or past students, your own previous submissions if you are repeating the course, etc.)*. We test every submission for copying and when we find it we treat it as flagrant academic dishonesty on the part of ***all*** parties involved, regardless of their roles as provider or consumer. The penalty is generally instant failure in the course, in addition to the usual sanctions applied by Student Judicial Affairs.

As you can see from the submission header below, we ask you to essentially take an oath that the code you submit is <u>***entirely your own work***</u>**.**

At the same time, we certainly understand very well that you will frequently need help completing your programming assignment, so we provide channels where you may get that help in a way which will strengthen your programming skills: instructor and TA office hours, and the discussion forums where you can ask all the questions you want about the assignment, and even help others with the understanding that you have gained *(but not, of course, with any actual code)*.

Assignment Submission Instructions

Your assignment must be submitted as a ***simple text file*** named ***main.cpp***
Files in ANY other format (MS Word document, etc.) or with ANY other name (main, main.doc, main.txt, etc.) will ***not*** be graded.

Submit your work via the appropriate iLearn assignment link, making sure you use the correct link and <u>**click on the attach button**</u>

We strongly recommend that:

1. you submit at least <u>*6 hours*</u> before the deadline, even if you haven't completed the program - you can re-submit as often as you like, and we will only grade the final submission
2. once you have submitted your final attempt, go back and download it back to your system: then run it just to make absolutely sure that it really is the file you intended to submit!

You are budding professionals, and are expected to take full responsibility for abiding by the requirements of a contract.

The ***only reason we will ever accept*** for a missed deadline is if the system administrators inform me that either the iLearn system or the CS department servers were off-line for a significant period around the time of the deadline *(In which case we will probably have notified you of alternative procedures)*.

Remember to include the following header information at the top of your program
(DO NOT REMOVE THE // at the beginning of each line, these tell the compiler to ignore these lines)

```
// Course: CS 10 <quarter & year>
//
// First Name:
// Last Name:
// Course username: <enter the username you use to login in the lab>
// Email address: <enter your cs or UCR student email address here>
//
// Lecture Section: <e.g. 001>
// Lab Section: <e.g. 021>
// TA:
//
// Assignment: <assn1, hw2, lab3, etc.>
//
// I hereby certify that the code in this file
// is ENTIRELY my own original work.
//
// =================================================================
```

NOTE: This header MUST appear at the top of EVERY file submitted as part of your assignment
(don't forget to fill in *your* details and remove the <> brackets!!).

Copy & paste this header into your file then update the personal details.

Assignment Specifications

For this assignment you will create a madlib, constructing a story around several user inputs. Search Google for madlibs if you are not familiar with this concept. You are to come up with your own madlib, so let your imagination go wild!

Additional Knowledge

Examples of madlibs: www.madglibs.com or www.wordlibs.com

Your Assignment

You do not need to create your own story, but do not just copy one of these madlib examples. Find or create a short story of your own and then convert it to a madlib similar to the examples above.

However, to ensure you learn from this exercise, your madlib must have the following:

- You must have a minimum of **10 inputs** from the user. These inputs should all be stored as strings. However, a single input must not be more than 1 word (or 1 number).
- Your output must have at least **3 paragraphs**, each paragraph **separated by a blank line**. Each paragraph must have at least **3 sentences**.
- Your output must have complete sentences with proper punctuation, grammar, and spacing around words.
- No line of output should have more than 80 characters.
    - When testing your code we will never input more than 1 word answers.
    - We will always use inputs of 10 characters in length.

Take these things into consideration when determining where to put a newline character in your output so the output never has more than 80 characters in a single line.

With the 10 character inputs in the variable output locations, the output should try to get as close to the 80 character line limit as possible without exceeding it. This means you should not have a line of 60 characters, as one or more words more than likely could have fit on that line without going over 80. This rule does not apply to the end of a paragraph.

Formatting Help

Every line (except the last in a paragraph) must be as close to 80 characters long as possible (when all input words are 10 characters each). We test this by looking for both "too long" lines and "too short" lines. Here are some tips on how get the line length right:

Check your output line length in comparison to terminal window size:

- If you open a terminal window (either when in the lab or in NX [gnome interface]) you can grab the right window border of a terminal and drag it to be 80 characters. A small tooltip pops up to show the width and height. If you use a Mac then we believe it is displayed in the topmost window dressing. Geany's execution window is 80 characters by default.
- Test your formatting with 10-character words: Each line should be as close to 80 characters as possible (except for the last line of a paragraph, of course), and never exceed 80.
- Paragraphs should have a blank line between them.

Viewing the Result
Run your program (again, with 10 character inputs) in your 80 character wide terminal (see above). Then grab the side of your terminal and expand it out to 100+ characters: it should still look the same - i.e. each line should still be no more than 80 characters.

Scoring Issues
When a story contains all long lines or all short lines, the resulting score will be a 0 for formatting as you cannot get points for formatting by not doing it at all!

Turn-in Reminders

Your submission should be have an attached file named `main.cpp`. (`Main.cpp` is not acceptable.) Verify the file contents after submission.

Marking Guidelines

- Deductions: incorrect header, compilation failure or file name not `main.cpp`
- Correctly declares at least 10 variables.
- Correctly obtains at least 10 inputs from user.
- Output has at least 3 paragraphs each separated by a blank line.
- No line of output has more than 80 characters.
- Lines of output attempt to get close to 80 characters (non paragraph ending lines)
- Use complete sentences, proper punctuation, grammar, and spacing around words

Basic Style Guidelines

- **indentation** - all lines inside the main function should be indented 2-4 spaces
- **spacing** - blank lines should be used to separate logic blocks
- **no line wraps** - no line of code should have more than 80 characters
  (this is distinct from the requirement that a line of **output** cannot be more than 80 characters)
- "**meaningful**" variable **names**
- **Comments** - each logical block of code should have a brief explanatory comment.