Assignment 5

Morse Code Translator

Due: Sunday, February 28th, 2016 11:59pm

Grading: EVERY assignment in this course is graded by demoing your work for 10 minutes with a TA. You are required to meet with a TA within one week from the due date to demo. You are penalized for failure to see a TA within the week or missing a scheduled appointment. In either case, if you are within 1 day (24 hours) of the deadline, you lose 10 points. If you are within 7 days (1 week) of the deadline, then you lose 25 points, anything outside of a week from the deadline to demo is an automatic 50 point deduction. Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program☺

Implementation (70 points)

For this assignment, you will write a program that allows the user to enter a message by inputting morse code that you will translate to text OR text that you will translate to morse code. In the last assignment you learned how to use C++ strings-- you're **not allowed to use any C++ strings in this program.** Instead, you must use C-style strings, which are stored in an array of characters, ended by a null character, '\0'.

What if the user doesn't know how big the sentence or word is? Plus, it is awkward to limit the user to a specific number of characters or to ask the user for the number of characters he/she is going to enter. Don't you want them to just enter something, and then an array will be created just big enough to store their sentence, like the C++ string?

You will create your C string by reading characters from the user until you see a newline, '\n'. Since you want to consume the newline character to know when to stop growing your array on the heap, then you will need to use the get() function, i.e. cin.get()**.** You can read more about this function and how to use it:
http://www.cplusplus.com/reference/istream/istream/get/
Remember, this function will no longer append the '\0' to your string, it is up to you to make sure this is the last character in your c-style string!!!

The **cstring** library has a variety of functions that can help you work with C-style strings, but unfortunately you're **not allowed to use any of those, either.** If you see a function you want to use in the cstring library, you need to figure out how to implement it yourself! (Don't worry-- the most useful ones aren't complicated to recreate.)

Same rules as in Assignment #4, you must have function <= 15 lines of code (-10 pts, otherwise) and NO global variables (-10 pts, otherwise)!!!

Please refer to the following document for help with morse code:
http://morsecode.scphillips.com/morse2.html

Be creative with how you will do the translation from one to another using this chart! Notice, you only have to worry about one case and only a few punctuations.

Morse code input and output rules:

- Three spaces between characters.
- Seven spaces between words.

Here is an example of your program:

Do you want to translate text to Morse code (1) or translate Morse code to text (2)? **1**
Enter Text Message: **Hello World**
Morse Code:

```
....   .   .-..   .-..   ---       .--   ---   .-.   .-..   -..
```

Do you want to translate text to Morse code (1) or translate Morse code to text (2)? **2**
Enter Morse Code:

```
....   .   .-..   .-..   ---       .--   ---   .-.   .-..   -..
```

Text Message: **Hello World**

(10 pts) Extra Credit Error Handling

Make it so that your program never errors on users input! Handle the following errors:

- The user enters a message or letter outside of A-Z, a-z, and some punctuation.
- The user enters morse code with something other than . and – in it.
- The user enters an invalid menu choice.

(10 pts) Design

You will design a solution for the problem statement as well as answer the following questions for **each** function that you write.

- What is being passed into this function?
- What is being returned?
- What must be true **before** this function can be called?
- What will always be true **after** this function is called?

Your design can be in the form of a flow chart, pseudocode, or both. **DESIGN BEFORE YOU CODE.** While there is no way that we can check that you actually did, it will make writing the code take much less time.

(10 pts) Testing

Fill out this testing table for your program (the first row is an example and you must test more than one). Remember, you need to have good and bad input values, even if you don't handle the errors!!!

Inputted Values	Expected Output	Actual met Expected
he	`....  .`	yes

(10 pts) Program Style/Comments

In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!
http://classes.engr.oregonstate.edu/eecs/winter2016/cs161-001/161_style_guideline.pdf

```
/*************************************************
** Program: morse.cpp
** Author: Your Name
** Date: 2/20/2016
** Description:
** Input:
** Output:
*************************************************/
```

Electronically submit your C++ program (.cpp file, not your executable!!!) and pdf, by the assignment due date, using TEACH.

https://secure.engr.oregonstate.edu:8000/teach.php?type=want_auth

**NOTE: The easiest way to upload your program from ENGR to TEACH is to map a network drive to your home directory on ENGR. Mac or Windows, See:

http://engineering.oregonstate.edu/computing/fileaccess/

If you are doing this off campus, pay attention to the off-campus directions!!!!