PIC 10C Section 1 - Homework # 9 (due Monday, November 29, by 11:59 pm)

You should upload each .cpp and/or .h file separately and submit them to CCLE before the due date/time! Your work will otherwise not be considered for grading. Do not submit a zipped up folder or any other type of file besides .h and .cpp.

Be sure you upload files with the precise name that you used in your editor environment otherwise there may be linker and other errors when your homeworks are compiled on a different machine.

AT THIS POINT YOUR CODE MUST COMPILE AND OPERATE UNDER G++ WITH O3 OPTIMIZATION AND C++2a STANDARDIZATION!!!

ONE WAY MESSENGER APP

In this assignment, you'll write C++ code that when compiled and linked, produces an executable that can give live updates from a file, thus serving as a primitive messenger service. Specifically, your programs should run on the PIC servers with a live website (don't worry, you'll be given all the web programming code and you only need to write C++). When someone visits the website, if the C++ program is running, then messages they send through the site will be displayed in the C++ console.

Remark: there's nothing special about the files being part of a live website. As you debug, you could even write/edit a file that is local to your executable to test that when the file is updated, the message changes. The novelty here is really that you can hopefully see how C++ programs can be integrated into more complex systems, even involving other languages.

You should submit the files **messenger.h**, **messenger.cpp**, and **main.cpp** and you can assume your homework will be built beforehand with the makefile:

```
msg.out: messenger.h messenger.cpp main.cpp
    g++-8 -std=c++2a -O3 messenger.cpp main.cpp -o msg.out -pthread
```

It is very important you follow the instructions for making a webpage stated at the end of this document to a tee!

You will need to write a **message** class as outlined in the assignment. Otherwise, the rest of the implementation is up to you. Please refer to the syllabus for how the work will

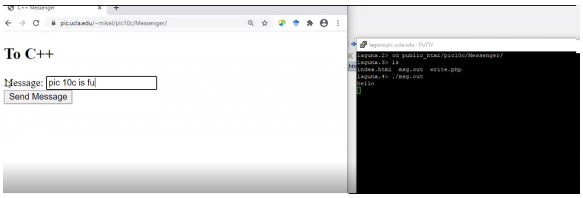

Figure 1: Before a user submits on the website, the message does not display...

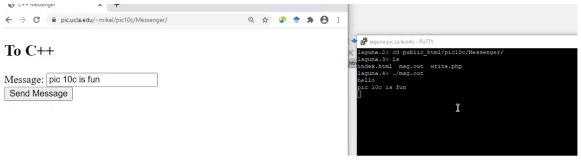

Figure 2: Once the user on the website submits, the message appears.

be graded: note that more than half of the marks come from good coding practices and code documentation.

The **message** class shall:

- have a single member variable, **msg**, storing the current message;
- have a member function **set_msg** that (1) reads from a file "to_cpp.txt" character-by-character into a **std::string** and assigns that new value to **msg** then (2) waits 1 second before checking for an update and repeating;
- have a member function **get_msg** that returns the value of **msg**; and
- have a single default constructor that invokes **set_msg** in a **detach**ed thread.

The program should operate such that every time there is a new message, that message is displayed in the console window (possibly with a 1 second delay). Additionally, the program stays running until the user types 'q'. Screen shots are included for your reference. You can also see a quick demo on **messenger_demo.mp4**.

Hints:
Besides the **message** class creating a thread, the task can be accomplished with two additional threads: (1) a thread that "listens" for the user to press 'q' by reading **std::cin.get()** and (2) a thread that prints the messages when they change. You may even be able to wrap the (2) functionality into the **message** thread. The threads will need a signal for when to terminate: this can be when the user presses 'q'.

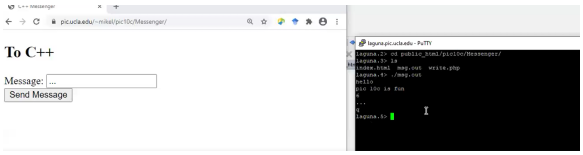

Figure 3: To quit, a user must press 'q' and then RETURN/ENTER.

WEBSITE INSTRUCTIONS

This isn't a web programming class so you'll need to follow these instructions on faith, without fully understanding them. All of this is covered in PIC 40A. You can also ask in Office Hours about this, but the steps here are beyond the scope of the course.

- In your home directory (after ssh-ing in), make a directory, **public_html**.
- Then type **chmod 755 public_html**
- Move into **public_html** and make a directory, **pic10c**.
- Then type **chmod 755 pic10c**
- Move into **pic10c** and make a directory, **Messenger**.
- Then type **chmod 755 Messenger**
- Move into **Messenger**.
- Transfer the two files given to you, **index.html** and **write.php**, into that **Messenger** directory without making any changes whatsoever to them.
- Then type **chmod 755 *** from within that **Messenger** folder.
- Type **dos2unix ***
- If you did things correctly, you should be able to visit **https://www.pic.ucla.edu/~your_username/pic10c/Messenger** and see the webpage (use your actual PIC username in the URL). If you enter a message and Submit it, you should see that the file **to_cpp.txt** is created / written to with the message contents.
- If that all works, now you can write your C++ program in that **Messenger** directory and test it out.