COMP2710: Homework 1

Due: September 10th, 2014 at 11:55 pm on the Canvas system

Goals:

To find someplace where you can develop, compile, and run C++ code.
To write a very simple C++ program and learn a few things about buffer management using C++ String class.

Questions (put your answers in a pdf or txt file):

The C++ string class is useful storing and manipulating with strings (texts). Answer the following questions that highlight some of the useful features of C++ strings. To verify if your answers are correct, you may write a small program to test them.

1) Write a code fragment that will use the "+" operator to append the string `username` and group name (e.g. "`All`" which means to all users) to the string `message` and store it into the string `message_buffer` where the username is first bracketed by the strings "`|<`" and "`>|`". The username and the group name are separated by the string "`::`". The strings are defined as follows:

```
string username = "George";
string message = "Join our Tigers message group!";
string message_buffer;
```

After executing your code fragment, `message_buffer` must contain the following string:

```
|<George::All>|Join our Tigers message group!
```

2) Write the code fragment that will use the above code in (2) to allow the program to prepend more user name, the group name (`Tigers`) and message into message buffer created in (2).

```
string username2 = "Abraham";
string message2 = "Good morning, Tigers!";
```

After executing your code fragment, `message_buffer` should contain the following string:

```
|<Abraham::Tigers>|Good morning, Tigers!|<George::All>|Join our Tiger message group!
```

3) Write the code fragment that will insert a '`\n`' character between two messages `message3` and `message4` and store them in `message_buffer`.

```
string message3 = "War Eagle";
string message4 = "Hey!";
```

After executing your code fragment, `message_buffer` should contain the following string:

```
War Eagle\nHey!
```

Print out this string in `message_buffer` and describe what happens.

4) The variables `username` and `message` in (2) are objects, which mean that they encapsulate both functions and attributes. There are many useful functions in the `string` class. What does the function `string.size()` return? For example, in question (2), what will `message_buffer.size()` return?
5) What will be the effect of executing the code `message.empty()` in question (2)?

*Questions **6**-11 refer to the state after the code fragment in **(2)** is executed.*

6) Write the string function call that will return the index of the *first* occurrence of a string "|<" in `message_buffer`.
7) Write the string function call that will return the index of the *second* occurrence of a string "|<" in `message_buffer`.
8) Write the code fragment that will return the string containing the *first* user name in `message_buffer`. (The user name must not contain "|<", ">|", "::" or the group name.)
9) Write the code fragment that will return the string containing the *second* user name in `message_buffer`.
10) Write the code fragment that will return the string containing the *first* group name in `message_buffer`. (The group name must not contain "|<", ">|", "::" or the user name.)
11) Write the code fragment that will return the string containing the *second* message in `message_buffer`.
12) Assume that a string buffer is defined as follows:

```
string messagebuffer;
```

What is the difference between the following codes, e.g. when the user input the line "Aubie Messaging Group"?

```
a. cin >> messagebuffer;
b. getline(cin, messagebuffer);
```

Program (submit your code listing and a sample script of the execution):

A very common application is virtual bulletin board (such as facebook, twitter, etc.), chatroom or texting on your cell phone or computer. Messages can be posted on the bulletin board and are readable from a group of users. Messages can also be sent from a user to another user or multiple users through the network. In some cases, conference call involving multiple participants can chat with each other in real time almost simultaneously. The applications that handle your messages must first buffer the messages before they are sent. There are many challenging problems that must be addressed for multiple users to exchange messages and access shared buffers concurrently. However, instead of solving these difficult problems, you will focus on how to manage message buffers.

The purpose of this homework is for you to write a short program in C++. Compile and run it using the g++ compiler on a Linux computer (either in Davies 123, your home Linux machine, using secureCRT to log into a remote tux Linux machine, or using a virtual machine like VirtualBox on your Windows PC). If you test your programs on your own Linux computer, you must eventually ensure that your program works correctly in the College of Engineering tux Linux computers in Davies 123 for full credits.

Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. You will lose points if you do not have a comment block on **EVERY** program you hand in. Also describe any help or sources that you used (as per the syllabus). Your program must be written in the COMP 2710 C++ programming style guide described in the handout.

Write a program that will continually prompt for the user name, group name and then prompt for multiple lines of text messages. (The user name and group name consists of only one word each.) The text message entry is terminated when in a new line, the user enters "`$$`" and the enter key, i.e. a line with only the string "`$$`". The string "`|<`" must NOT be included in the message. When the program reads the inputs, it will then store the user name and their message into a message buffer that is of the C++ `string` class. There is only one shared message buffer to store multiple user names and their messages. All messages must be prepended to the message buffer, i.e. the latest message must be placed before the older messages in the shared message buffer. In other words, the shared message buffer contains messages that are in *reverse chronological order*. The format for prepending the username and message into the shared message buffer is as follows. The user name must be followed by the group name, separated by the string "`::`" and then they must be enclosed by the strings "`|<`" and "`>|`"followed by the user message before pre-appending into the shared message buffer. Examples are shown in Questions (2) and (3) above.

After a user terminated their message, the program will then prompt if there is any more user name. If the answer is "`yes`", then it starts the cycle again. The name of the *next* user name and message will be prepended to the same shared message buffer. Note that the start of each user name and message can be detected by the string "`|<`". If the answer is "`no`", it will then proceed to the next stage of outputting the user names and messages stored in the shared message buffer. The output will display the text, "`The current messages are:`" and then must show all the user names and messages stored in the shared message buffer in ***reverse chronological order***, i.e. the latest message first. Each message must start on a new line, where the user name is printed followed by the group name in parenthesis and then the "`>>`" string and a newline and then the user message. Your program must print a blank line after each message.

Here is a sample dialog (where the user input is depicted as **Bold**, but you do not need to display user input in bold.):

```
============================================================
|          Welcome to the Auburn Messaging System!         |
============================================================

Enter user name> George
Enter group name> All
Enter the message> Join our Tigers message group!
It was fun at the beach.
How was your summer?
$$

Any more users? > yes

Enter user name> Abraham
Enter group name> Tigers
Enter the message> This is a great group.
Our Alaskan tour was amazing!
$$

Any more users? > yes

Enter user name> Kate
Enter group name> Tigers
Enter the message> Good to hear from you all.
Brazil was an interesting place.
$$

Any more users? > no


The current messages are:

Kate(Tigers)>>
Good to hear from you all.
Brazil was an interesting place.

Abraham(Tigers)>>
This is a great group.
Our Alaskan tour was amazing!

George(All)>>
Join our Tigers message group!
It was fun at the beach.
How was your summer?
```

Your program's output must match the style of the sample output.

Deliverables in .pdf or .txt format:

- Answers to the questions.
- A source code listing with instructions on how to compile your code (commands)
- A script of a sample run printed from the console.

Updated (AL): September 9, 2014 3:20pm