COMPUTER SCIENCES AND SOFTWARE ENGINEERING
AUBURN UNVERSITY

COMP 2710
Section 001
Software Construction

Fall 2017

Lab 1
TigerBook Social Network

Due: October 9, 2017

Points Possible: 100 (25 Design, 75 code)
Due:
Design Portion: **Wednesday, October 4th, 2017 by 11:55 pm** turned in via CANVAS SUBMISSION
Programming Portion: **Monday, October 9th, 2017 by 11:55 pm** turned in via CANVAS SUBMISSION

Goals:

- To develop some proficiency with basic C++ syntax and semantics.
- To organize code into classes and abstractions (ADT).
- To gain some familiarity with formal testing.
- To write a fun application!

Design Portion:
Create a text or PDF file named "<username>_1w.txt" (for example, mine would read "lim_1w.txt"). Please submit a text (.txt) file or PDF (.pdf), *do not submit MS Word Documents or other proprietary formats*.

Process (steps 1-3 due with initial turn-in, steps 4-5 due with final turn-in:
Each of the following should address the program specification below. Concern yourself more with thinking about the problem than worrying about the specifics of implementation at this stage. For the first turn-in, you will provide steps 1-3. For the final (electronic) turn-in, you will provide all the steps (including any revisions you may have made). Please note revisions in your final copy! You should provide:

1. **Analysis (10):** Write a few important **use cases**. Remember, these describe how a user interacts with the system (what they do, what the systems does in response, etc.). These need not be long, but they should have enough basic details such that someone unfamiliar with the system can have an understanding of what is happening. They should not include internal technical details that the user is not (and should not) be aware of. Make sure that any special rules/features you plan to add are clearly described in your Analysis section.

2. **Design (10)**: From your use cases, identify likely candidates for software objects, the processes related to the objects and the data stores.
   a) Create a Class Diagram. The class descriptions must include:
      1) The name and purpose of the class
      2) The member variables and the functions of the class
      3) Show the interactions between classes (for example, ownership, use or dependency)
      4) Any relevant notes that don't fit into the previous categories

   b) Create the data flow diagrams to describe the system. Show all the entities and processes that comprise the overall system and show how information flows from and to each process.

3. **Testing (5)**: Develop lists of specific test cases:
      1) For the system at large. In other words, describe inputs for "nominal" usage. You may need several scenarios. Also, suggest scenarios for abnormal usage and show what your program should do (for example, entering a negative number for a menu might ask the user to try again).
      2) For each object. (Later, these tests can be automated easily using a simple driver function in the object).

Remember, test cases are specific, repeatable, and should have a clearly defined result.

4. Implement your solution in C++ (70).

5. **Test Results (5)**: ***After developing your solution***, actually try all of your test cases (both system and unit testing). Actually show the results of your testing (a copy and paste from your program output is fine – don't stress too much about formatting as long as the results are clear). You should have test results for every test case you described. If your system doesn't behave as intended, you should note this.

Program Portion:

One of the most commonly used application in the smart phones, laptops and desktops is the Facebook social network for sharing information, messaging or texting application. In this Lab 1, you will analyze, design, implement and test a simple message sharing application, called TigerBook Social Network, which *implements new capabilities that are absent from Facebook* and other social networks. It will have a simple text-based user interface that allows multiple users to share messages through one of the three methods: (1) **broadcast** a message to all users, (2) **multicast** a message to a group, or (3) **unicast** a message to a particular user. In this lab project all the users and the message buffer must be implemented in the *same address space*, i.e. the message buffer memory space must be accessible by all the users. Your program must allow a user *U* to display in their **home page** all multicast messages to each group *G* where *U* is a member of, all broadcast messages, and all unicast messages to *U*. Your program must allow a user *U* to display in their **wall page** all messages posted by *U*.

What is the TigerBook Social Network?

The concept of TigerBook Social Network is similar to *Facebook* that is widely used for social networking and sharing of information among friends. The exception is that for the purpose of this class assignment, we will simplify it to deal with only text messages; so no images or videos will be posted. This project also extends the capabilities of *Facebook* with group communication facility. For this assignment's purposes, the TigerBook Social Network is a program (or collection of programs) that does the following:

- There are many users that use the TigerBook Social Network for posting their own messages and sharing them with their friends. Each user posts their message using either broadcast, multicast or unicast communication method. All messages, including broadcast, multicast and unicast messages sent by any user will be stored in the same message buffer. *In all cases, the program must allow multiple lines of a message to be posted by each user.* In this assignment your implementation must handle **at least four users**.
- When a user ***broadcast*** a message, it will be seen by all users. You can use the group name "`#All`" to be associated with each broadcast message. *The program must allow multiple lines of a message to be broadcast by each user.* All users will display broadcast messages in their home page.
- ***Multicast*** messages allow users to share messages with only the users who are members of a particular group. When a user multicast a message, the system will prompt for the group name. Group names always start with a '#' character, e.g. "#Tigers". When storing a multicast message in the `message_buffer`, the sender's name, the *group* name and the message must be stored using the specified format. For instance, when `Kate` sends a multicast message to the group "`#Tigers`", the `message_buffer` may contain the following: `<!Kate:#Tigers!>War Eagle!` The system must check if the group name is valid. Only users who are members of a group will display the group message in their home page.
- You may use a group manager will maintain the different groups used in the system, including creating and deleting a group, adding users into a group and removing users from a group. When adding a user in the group, the system must check that the user to be added is a valid current user.
- When a user ***unicast*** a message, it will be seen by only the recipient user. The system will first prompt for the recipient user name. When storing a unicast message in the `message_buffer`, place the recipient name in place of the *group* name. For instance, when `Joe` sends a unicast message to `Mary`, the `message_buffer` may contain the following: `<!Joe:Mary!>Good Morning, Sunshine!` Only the recipient will display the unicast message in their home page.
- Each user can display their home page. The **home page** of a user displays all the messages that are received from broadcast, multicast and unicast messages and all of their own messages, in reverse chronological order in which they are created, i.e. the most recent messages are displayed before the less recent messages. When displaying the messages, you must display the group name for multicast, recipient name for unicast or "`#All`" for broadcast. For multicast messages, the user must be a member of the group. For unicast message, the user must be the recipient.
- Each user can display their wall page. The **wall page** of a user displays all the messages that are sent by the user, including broadcast, multicast and unicast messages, in reverse chronological order in which they are created. When displaying the messages, **you must** display the group name for multicast, recipient name for unicast or "`#All`" for broadcast; followed by the messages.

- In the home or wall pages, the program will first display only the two most recent messages (multicast, unicast, or broadcast) from other users or sent by themselves, and prompt if the user wants to display more messages. If the response is "yes", then all messages from themselves and all other users and will be displayed, otherwise, the program will stop the display.
- Since this assignments implements a simple interface, TigerBook Social Network will allow only one user to be logged in at a time; so it will allow for change of user. When a user is logged in, all operations will be for the current user.

Message format

For this assignment, the messages and user names must be stored in the following format, similar to the format used in Homework 1. There must be only *one* message buffer to store all messages from all the users, placed in a *reverse* chronological order from which they are created, i.e. all messages must be **prepended** to the message buffer. In other words, ***the latest message must be placed before the older messages in the shared message buffer.*** Each message must contain the user name, group (or recipient or "#All") name and their message. The username and the group (or recipient or "#All") name are separated by the string `":"` and then the resultant string is enclosed by the strings `"<!"` and `"!>"` followed by the user's message. Different lines of a message must be separated by a '`\n`' character. For examples, if `George` first broadcast the message `"Join our Tigers message group!"` and later, when `Kate` multicast a message to the #Tigers group, `"Good to hear from you all!"`, then the message buffer must contain `"<!Kate:#Tigers!>Good to hear from you all!<!George:#All!>Welcome to the group!"`. Each message must be of variable length. The purpose for this standard message format is that all users should end up being compatible. That is, in a distributed environment in future, every user should be able to work with other users' messages.

Your program must meet the above requirements for internal message format, otherwise significant points will be deducted.

The user interface

Write a menu-based and text-based user interface for interacting with the TigerBook Social Network. The main menu has (at least) 10 options to choose from as shown below. *You **must** abbreviate the menu so that all the 10 options can be shown in only two or three lines.* The user interface will accept the option and all related inputs from the user, perform the operation and then repeatedly shows this menu of options again and prompts for the next option.

- **Create a new user:** When selected, the program will then prompt for the name of the user and insert the name in one of the user objects. When a user is created, that user becomes the *current user*. The program must prompt a welcome banner, e.g. saying `"Welcome to TigerBook Social Network, Joe!"`, which should be centered on the screen and surrounded by a box.
- **Broadcast a message**: When selected, the program will prompt for the message and stores the message in the message buffer. *Your program must allow multiple lines of messages. To end a message, the user will enter a new line with a string `"^!"`followed by the enter key. The final string `"^!"`must not be stored in the message buffer, neither should it be displayed in the wall or home pages.*
- **Multicast a message**: When selected, the program will first prompt for *group* name and then it prompts for the message. As in broadcast messages, your program must allow multiple lines of messages. To end a message, the user will enter a new line with a string `"^!"`followed by the enter key.
- **Unicast a message**: When selected, the program will first prompt for *user* name and then it prompts for the message. As in broadcast messages, your program must allow multiple lines of messages. To end a message, the user will enter a new line with a string `"^!"` followed by the enter key.
- **Display wall page**: When selected, the program will first display a title indicating that it is displaying the current user's wall page, e.g. "`Joe's Wall Page`". It then displays the *two* latest messages in the current user's wall page, in reverse chronological order. Since the wall page contains only messages from the current user, *the wall page must NOT display the user names*; instead it only displays the user's messages and the group (for multicast) or recipient (for unicast) or "#All" (for broadcast) in reverse chronological order. Messages from different posting must be separated by a blank line. After displaying the two latest messages, it will then prompt the user if they want more messages. If the response is "no", then it will stop displaying messages, but if the response is "yes", it will display all the remaining messages from that user. If there are two or fewer messages then the program must not prompt for more messages.
- **Display home page**: When selected, it first displays the current user home page title, e.g. "`Joe's Home Page`", and then it displays only the two latest messages (either from broadcast, multicast or unicast messages), whichever are the two latest messages. When displaying the message from each user, it must display the sender name followed by the group name (or #All or recipient name) in parentheses and followed by a string " > " and a '\n' and then followed by the message. Messages from different users must be separate by a blank line. After displaying the latest two messages, it will then prompt the user if they want more. If the response is "no", then it will stop displaying messages, but if the response is "yes", it will display all the remaining (multicast, broadcast or unicast) messages for that user. If there are two or fewer messages then the program must not prompt for more messages. In both the

cases above, the messages are all displayed in reverse chronological order, i.e. the most recently posted messages will be displayed before the earlier messages.

- **Create a group**: When selected, the program will then prompt for the name of the group. It then checks if the group's name is already an existing group. If so, it will display an error message and prompts for another group name.
- **Join a group**: When selected, the program will first prompt for the name of the group. It then checks if the group's name is already an existing group. If not, it will display an error message and prompts for another group name. Next, the program will add the current user name to the group.
- **Switch to a different user**: When selected, the program will prompt for the user name, e.g. Jane, and then checks if the name is a valid user, i.e. has been created. If so, it switches current user to Jane. If it is not a valid user, it displays the error message and repeatedly asks for the user name. (Your program must not exit or terminate when this naming error occur.) When switched to a different user, the program must prompt a welcome back banner, e.g. saying "Welcome back to TigerBook Social Network, Jane", which should be centered on the screen and surrounded by a box.
- **Quit the** TigerBook Social Network**:** When selected, the program will exit gracefully.

The user interface must check for correct input value from the users. If there is any error, e.g. invalid user name, then the program must display the error and continue to prompt for the correct input. Your program must not crash, exit or terminate when there is an incorrect input value.

The name of your program must be called <username>_1.cpp (for example, mine would read "lim_1.cpp"

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 use the specific program file name, or do not have a comment block on **EVERY** program you hand in.

Your program's output need not exactly match the style of the sample output (see the end of this file for one example of sample output).

Important Notes:

You must use *an object-oriented programming* strategy in order to design and implement this TigerBook Social Network application (in other words, you will need write class definitions and use those classes, you can't just throw everything in main()). A well-done implementation will produce a number of robust classes, many of which may be useful for future programs in this course and beyond. Remember good design practices discussed in class:

a) A class should do one thing, and do it well
b) Classes should NOT be highly coupled
c) Classes should be highly cohesive

Some potential classes:

1. A ***Menu*** class which handles basic user screw-ups (choosing an option out of bounds).
2. A ***System*** class which instantiates the other objects that must be initialized.
3. A ***User*** class which maintains information on the user name, group name and wall page.
4. A ***Group*** class that maintains the members of the group.
5. A ***MessageBuffer*** class that maintains the message buffer, i.e. inserts and reads the sender, group, recipient and messages in the buffer.

You should at a very minimum have classes to handle menus, users and systems. Use your judgement to define additional classes or not use some of the classes above.

- You must follow standard commenting guidelines. Follow the C++ programming style guideline that was handed out.

- You DO NOT need any graphical user interface for this simple, text-based application. If you want to implement a visualization of some sort, then that is extra credit.

Error-Checking:

You should provide enough error-checking that a moderately informed user will not crash your program. This should be discovered through your unit-testing. Your prompts should still inform the user of what is expected of them, even if you have error-checking in place.

Submit your program through the Canvas system. If for some disastrous reason Canvas goes down, instead e-mail your submission to TA – Tianhang Lan – at (tzl0033@tigermail.auburn.edu). Canvas going down is not an excuse for turning in your work late.

You should submit the two files in digital format. No hardcopy is required for the final submission:

<username>_1.cpp
<username>_1p.txt (script of sample execution, especially the results of testing)

Sample Usage:

Bold letters indicate user's input.

```
          ============================================================
          |               The TigerBook Social Network!                |
          ============================================================

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Choose an option: n

Please enter user name: George


          ============================================================
          |        Welcome to TigerBook Social Network, George!      |
          ============================================================

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: g

Please enter the group name: #Tigers

          ============================================================
          |                  #Tigers group created                   |
          ============================================================

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: j

Please enter the group name: #Tigers

          ============================================================
          |               George is in #Tigers group                 |
          ============================================================

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: b

Enter message: Join our new #Tigers message group!
It was fun at the beach.
How was your summer?
^!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)
```

Please choose an option: **n**

Please enter user name: **Mary**

```
          ============================================================
          |          Welcome to TigerBook Social Network, Mary!      |
          ============================================================
```

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **j**

Enter the group name: #**Tigers**

```
          ============================================================
          |               Mary is in #Tigers group                   |
          ============================================================
```

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **m**

Enter the group name: #**Tigers**

Enter message: **Welcome back, guys!**
We had so much fun at Ireland!
^!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **n**

Please enter user name: **Abraham**

```
          ============================================================
          |         Welcome to TigerBook Social Network, Abraham!    |
          ============================================================
```

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **j**

Enter the group name: #**Tigers**

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: u

Enter the recipient user name: **Mary**

Enter message: **Good Morning, Sunshine!**
When did you get back from Ireland?
^!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **g**

Please enter the group name: #**Eagles**

```
          ============================================================
          |                  #Tigers group created                   |
          ============================================================


Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: j

Please enter the group name: #Eagles

          ============================================================
          |               Abraham is in #Eagles group                |
          ============================================================

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: b

Enter message: Just created a new Eagles group!
Come one, come all!
Have a blast!
^!


Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: m

Enter the group name: #Eagles

Enter message: Eagles have landed!
War eagle, hey!
^!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: m

Enter the group name: #Tigers

Enter message: I'm enjoying your group too.
Spent my summer in South Africa; it was amazing!
^!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: n

Please enter user name: Kate

          ============================================================
          |         Welcome to TigerBook Social Network, Kate!       |
          ============================================================

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g),
Join a group (j), Switch user (s), Quit (q)

Please choose an option: j

Enter the group name: #Eagles

          ============================================================
          |                 Kate is in #Eagles group                 |
          ============================================================
```

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **m**

Enter the group name: #**Eagles**

Enter message: **It's time for Auburn Football!**
War Eagle!
^!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **h**

```
         ==========================================
         |             Kate's Home Page            |
         ==========================================
```

Kate (#Eagles) >
It's time for Auburn Football!
War Eagle!

Abraham (Eagles) >
Eagles have landed!
War eagle, hey!
More message? (yes/no): **yes**

Abraham (#All) >
Just created a new Eagles group!
Come one, come all!
Have a blast!

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

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **s**

Enter user's name: **Mary**

```
         ============================================================
         |    Welcome back to TigerBook Social Network, Mary!       |
         ============================================================
```

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **h**

```
         ==========================================
         |             Mary's Home Page            |
         ==========================================
```

Abraham (#Tigers) >
I'm enjoying your group too.
Spent my summer in South Africa; it was amazing!

Abraham (#All) >
Just created a new Eagles group!
Come one, come all!
Have a blast!
More message? (yes/no): **yes**

Abraham (Mary) >
Good Morning, Sunshine!

When did you get back from Ireland?

Mary (#Tigers) >
Welcome back, guys!
We had so much fun at Ireland!

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

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **w**

```
==========================================
|             Mary's Wall Page           |
==========================================
```

(#Tigers) >
Welcome back, guys!
We had so much fun at Ireland!

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **s**

Enter user's name: **Abraham**

```
==========================================================
|    Welcome back to TigerBook Social Network, Abraham!   |
==========================================================
```

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **w**

```
==========================================
|            Abraham's Wall Page         |
==========================================
```

Abraham (#Tigers) >
I'm enjoying your group too.
Spent my summer in South Africa; it was amazing!

Abraham (#Eagles) >
Eagles have landed!
War eagle, hey!
More message? (yes/no): **yes**

Abraham (#All) >
Just created a new Eagles group!
Come one, come all!
Have a blast!

Abraham (Mary) >
Good Morning, Sunshine!
When did you get back from Ireland?

Create new user (n), Broadcast (b), Multicast (m), Unicast (u), Wall page (w), Home page (h), Create new group (g), Join a group (j), Switch user (s), Quit (q)

Please choose an option: **q**

```
==========================================================
|         Thank you for using TigerBook Social Network     |
==========================================================
```