Distributed TigerBook Messaging System

Robert Skelton

COMP-2710

Lab 2

10.28.13

Part 1: Analysis

The following are the use cases for the Distributed TigerBook Messaging System:

Create new user: This only happens on startup. The user will be prompted to make an account when the program is initially launched, and cannot create a new user or log out while the program is running. To create a different user, you will have to quit the program or run the program on another computer.

Add post: Posting allows only friends to see the message. The user will write a message of any number of lines. If the message contains any restricted characters, which are “{<”, “>}”,"{[", and "]}", that line will be

rejected, but the user will still be able to submit more. When submitted, the message will appear on the user's wall and on the homepages specified by the posting style (i.e. posting or tweeting).

Add tweet: Tweets allow anyone to see your post. A tweet is public, and can be viewed by the user, the user's friends, and people who are not the user's friends. The user will write a tweet of any number of lines. The tweet is not limited to 140 characters. If the message contains any restricted characters, which are "{<", ">}","{[", and "]}", that line will be rejected, but the user will still be able to submit more later. When submitted successfully, the tweet will appear on the user's wall and on the homepages specified by the posting style (i.e. posting or tweeting).

Add friends: The user will input the name of another user, and that person will be added to their friends list. This means that when the current user views their home page, the messages of their friends will appear. Friendship status does not have to be mutual. If the input name is not found, then the program will display an error message and return the user to the menu. If there are less than 2 users, this option will produce an error message.

Display wall: When the current user selects this option, the two most recent of their own messages or tweets will be displayed. Their name will not

appear by both of these messages since it is their own wall page. After this, the program will ask if they want to see more. If the user answers yes, the rest of the applicable messages will be shown. If the user answers no, they will be returned to the menu. If there is only one post, only that post will be printed. If there are only two posts, the user will not be prompted to show more posts.

Display home page: When the user selects this option, the two most recent tweets or the two most recent messages of the user or the user's friends (or a mix of both) will be displayed in reverse chronological order. Names will be associated with each message. The program will then prompt the user whether they want to see more. If the user answers yes, the program will return all of the rest of the applicable messages. If the user answers no, the program will return the user to the menu. If there is only one post, only that post will be printed. If there are only two posts, the user will not be prompted to show more posts.

Quit: Select the option to quit, and the program will end.

Part 2: Design

The following will be the main classes that could be implemented into the program. Likely data values and functions will be listed within each class. Outside any specific class, there will be a vector that contains every instance of the User class, which is all the users that have used Distributed Tiger Book, and an integer to keep track of the time stamp, which is to keep all messages and tweets in correct chronological order.

· **User:** Each instance of this class will represent an individual user of the messaging system. There will only be one of these per individual program run, because you can not logout, create a new User, or switch users.

o The constructor for this class will take in the username as a string. There is also a default constructor, which should never be called.

o String name: This private string indicates the name of the user.

o vector friendsList: This class will have a vector that contains the name of all friends. By default, it will also contain the name of the user at the first position.

o Function getName: This function will return the name of the user, because the string name is private. There will be no setName function.

- Function addFriend: This function will intake a name and check through the totalUsers and friendsList vectors, and also uses the isUser function. If there is no User instance with that name, it will return an error. If the name is already in the friendsList, it will return an error. Otherwise, it will append the name to the friendsList. A banner is shown if the friend is properly added.

- Function welcomeBanner: This function takes in the user's name, and prints a custom welcome banner welcoming them to Distributed Tiger Book. This is only called at the beginning of the program.
- Function isUser: This function takes in a string for the name, and checks to see if the user exists in the allUsers vector. If the user exists in the allUsers vector, and is not already in the friendsList vector, then the user can be added to friendsList vector if called.
- Function postMessage: Adds a message to the users text file.
- Function postTweet: Adds a public tweet to the tweet.txt file.
- File filename.txt: This file is created every time a new User object is created. Each User will have their own text file to store their posts, and is read when the home or wall functions are called.
- Function home: Shows all posts by the users friends and public tweets by all users.

- Function wall: Shows the users personal wall posts, and does not show the user's name.
- String messageBuffer: Temporarily holds messages in the User class.
- Function menu: calls other functions using a switch case. The main menu, essentially.
- Function getTweets: Return the tweet.txt file as a string, used for adding to the tweet.txt and printing all tweets.
- Function getPosts: Gets all posts of a certain user. Accepts a userNameIn as a string parameter.
- Function addToAllUsers: Adds a userNameIn to the list of all users text file. Accepts a string parameter.
- Function getAllPosts: Returns all posts by everyone in a string. Used for printing out a home page.
- Function addToAllPosts: Add a post or tweet to all posts by everyone. This is stored in a text file.

- **System:** This class will contain various functions that are used in overall flow of the program for Distributed Tiger Book.
  - Function isUser: Checks if the user inputted exists in the current user's friendsList vector. If so, returns true. If the user does not exist or is

already in the friendsList, returns false.

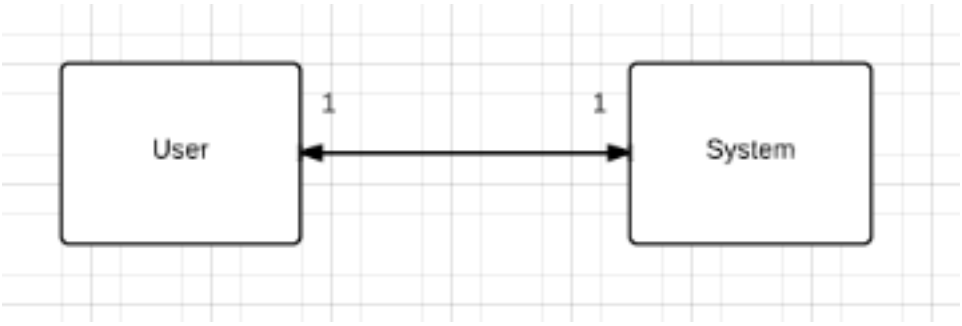
- The constructor of System takes no input and has nothing inside of it. It is there solely to create the new System object.
- Function getAllUsers: Returns the names of all Users who have registered for Distributed Tiger Book as a string.
- Function isUser: Checks if a user has registered with Tiger Book and returns a true or a false.

Not in a class:

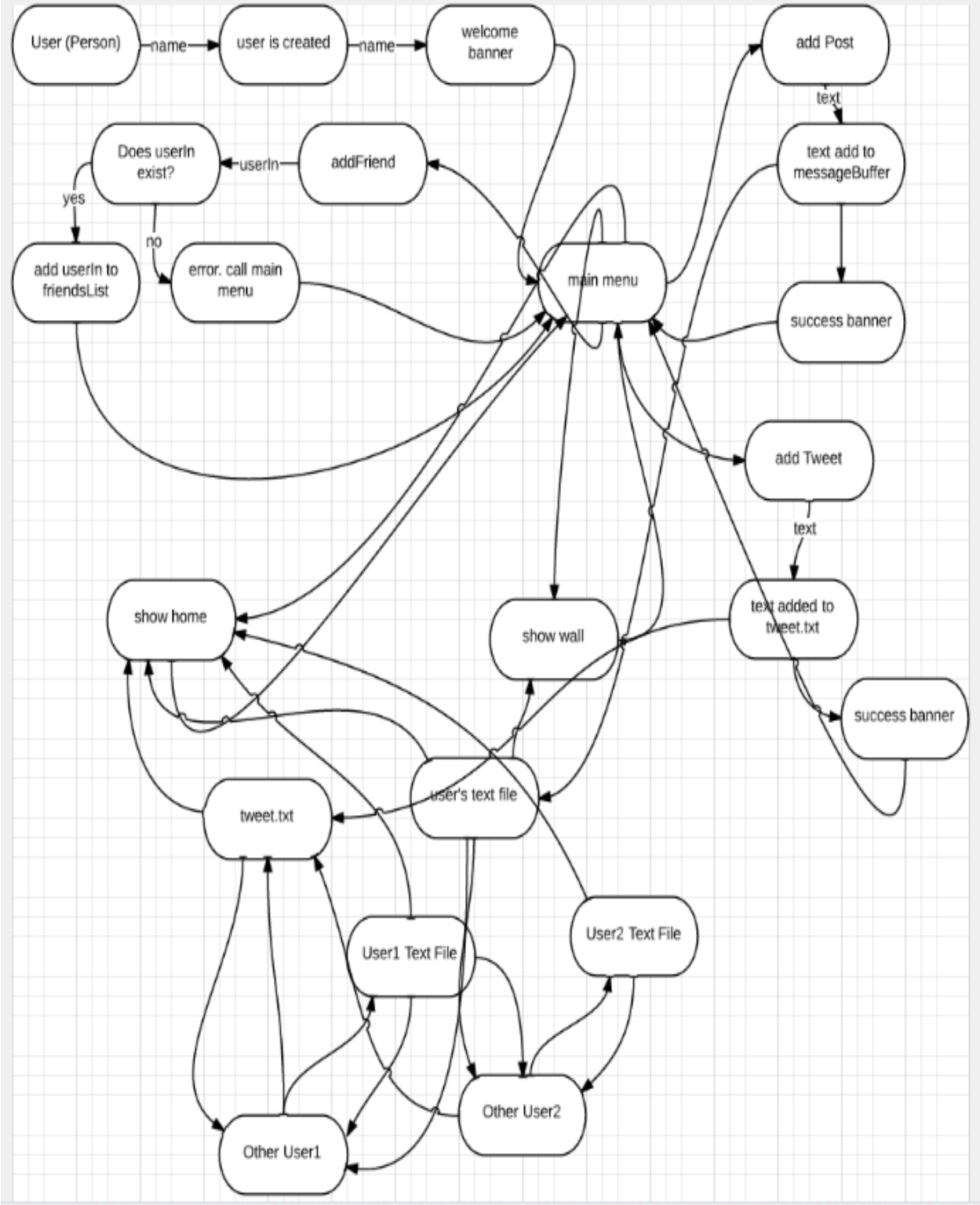
- Function setTimeStamp: Sets the time stamp to a given integer. This works together with incTimeStamp.
- Function getTimeStamp: Returns the int value of the time stamp.
- Function incTimeStamp: Calls getTimeStamp, then increments that by one, and then setTimeStamp to the new timestamp. This sets the new timestamp to one greater than it previously was.
- Function createUser: Creates a new User object by calling the User constructor, and prints a welcome banner.
- Function introBanner: Prints a banner when the program is started.
- Function welcomeBanner: Prints a custom welcome banner when a new User object is created.

- Function main: The main driver for the program.

The Class Diagram for my implementation of Distributed Tiger Book is shown below.



A basic Data Flow Diagram can be found below.



Part 3: Testing

The following are test cases are for the system at large. These test cases will be tested after I implement Distributed TigerBook in C++ programming language. For reference, the menu will be in this format:

Menu:

Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)

When the program is launched, a welcome banner is launched, and the user is prompted to enter their name. If the user name contains "{[", "]}", "{<", or ">}", creating the user will fail, and the person at the computer will be asked to choose a new user name. This is because of the way Distributed Tiger Book stores and parses through strings. After the user is successfully selected, another banner is launched that welcomes this unique user to Distributed TigerBook. Finally, a new User is created for this user, and the main menu is launched. This menu is the core of Distributed Tiger Book.

Comments in red are from my actual testing, after I finished the coding.

- **When selecting from the menu:** If menu selection input is not a lowercase or uppercase 'f', 'p', 't', 'w', 'h', or 'q', then an error will

be displayed and the user will be asked to choose again. So if the input is empty, an integer, or any letter or string besides ‘f’, ‘p’, ‘t’, ‘w’, ‘h’, or ‘q’, the user will prompted to enter another option. I decided to leave it as the first letter is parsed only. So if you type “post”, it will only see “p”, and it will call up the postMessage function. On the other hand, if you type “What”, it will call “W”, or showWall.

- **When entering a message:** When a user types ‘p’ or ‘P’, a prompt will be brought up to enter a message. This message will be stored in the current user’s messageBuffer, as long as it satisfies a condition. If the string contains the characters “{[”, “]}”, “{<”, or “>}”, an error will be issued and the menu will be called. This is because the way Distributed Tiger Book parses through the messageBuffer, those characters are used, and would mess up message output. A blank string will suffice, and will just add a blank message to the messageBuffer. No big deal. You’ll just look stupid to your friends for posting an empty message. Also, after every message is successfully added, the timeStamp will be incremented. A banner is shown if a message is added successfully. You can input any message you want as long it doesn’t contain any of those combinations of brackets.

Typing in “!!” will end the post, and the “!!” will not be stored as part of the post.

- **When adding a friend:** The user will type a ‘f’ or ‘F’ to add a friend. The user will then be prompted to enter a user name of the friend they want to add. If the user exists, that friend will be added to the current user’s friendsList vector. If the user does not exist, an error will be issued saying that the user does not exist, and the menu will be brought back up. If the user that is to be added exists, but is already in the current user’s friendsList vector, a message portraying that will be issued, and the menu will be called. Anything can be typed into the friend to add section, because a user’s name can be any string. In my implementation, I did not allow the “{[“ or “]}” brackets to be used in a user name. You can still type those when adding a friend, but there will never be any users that contain those character combinations in their user name. Otherwise, this part works as intended.
- **When entering a tweet:** When a user types ‘t’ or ‘T’, a prompt will be brought up to enter a public tweet. This tweet will be stored in the public file called tweet.txt, as long as it satisfies a condition. If the string contains the characters “{[”, “]}”, “{<”, or “>}”, an error will be issued and the menu will be called. This is because the way

Distributed Tiger Book parses through the tweet.txt, those characters are used, and would mess up the output when printing a tweet. A blank string will suffice, and will just add a blank tweet to the tweet.txt file. No big deal. You'll just look stupid to your friends for posting an empty tweet to the world. Also, after every tweet is successfully added, the timeStamp will be incremented. A banner is shown if a tweet is added successfully. You can input any tweet you want as long it doesn't contain any of those combinations of brackets. Typing in "!!" will end the tweet, and the "!!" will not be stored as part of the tweet.

- **When showing the home page:** When typing a 'h' or 'H' the user should be shown the home page. This starts with displaying the two most recent posts by friends or tweets by anyone, and showing their user name. That way you can know who posted that respective message or tweet. The user will then be prompted to know if they want to see more messages. The acceptable inputs will be 'yes' or 'no', in all forms of uppercase and lowercase. If the user types any other string, character, or integers, the program will just go back to the menu. After that, if the user wants to see the rest of the home page's messages and tweets, the user will need to ask to show the home page

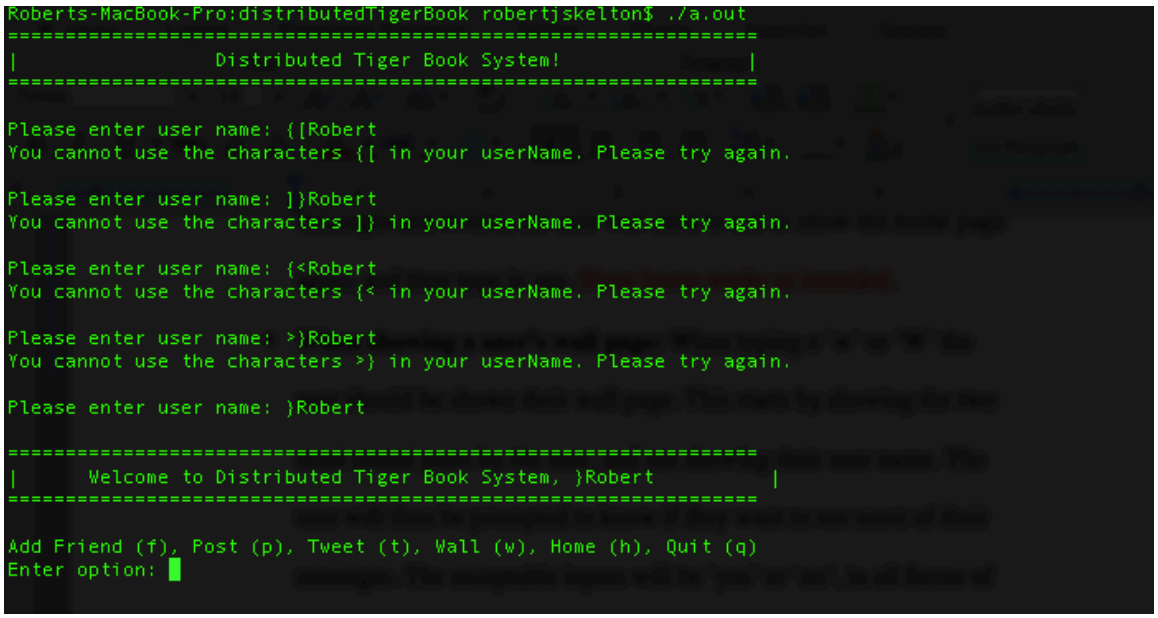
again, and then type in yes. Show home works as intended.

- **When showing a user's wall page:** When typing a 'w' or 'W' the user should be shown their wall page. This starts by showing the two most recent posts by the user, and not showing their user name. The user will then be prompted to know if they want to see more of their messages. The acceptable inputs will be 'yes' or 'no', in all forms of uppercase and lowercase. If the user types any other string, character, or integers, the program will just go back to the menu. After that, if the user wants to see the rest of their wall page's messages and tweets, the user will need to ask to show the wall page again, and then type in yes. Show wall works as intended, and never prints out the current user's name. The user's name is only shown at the banner at the beginning and end of the show wall call. Rhymes all day.
- **When quitting:** When a user types 'q' or 'Q', the program will end. No ifs, ands, or buts. The txt files for the user and the tweet text file will remain in the server's memory until they are overwritten when the program is run again. Quit works as expected. This one gave me some trouble for a while, because I was calling the menu in createUser and at the end of every other function also, so I would have to type in "q" twice to quit the program. I ended up taking out the

menu call from createUser and pasting it into main instead, and that fixed my issue.

Specific Test Cases

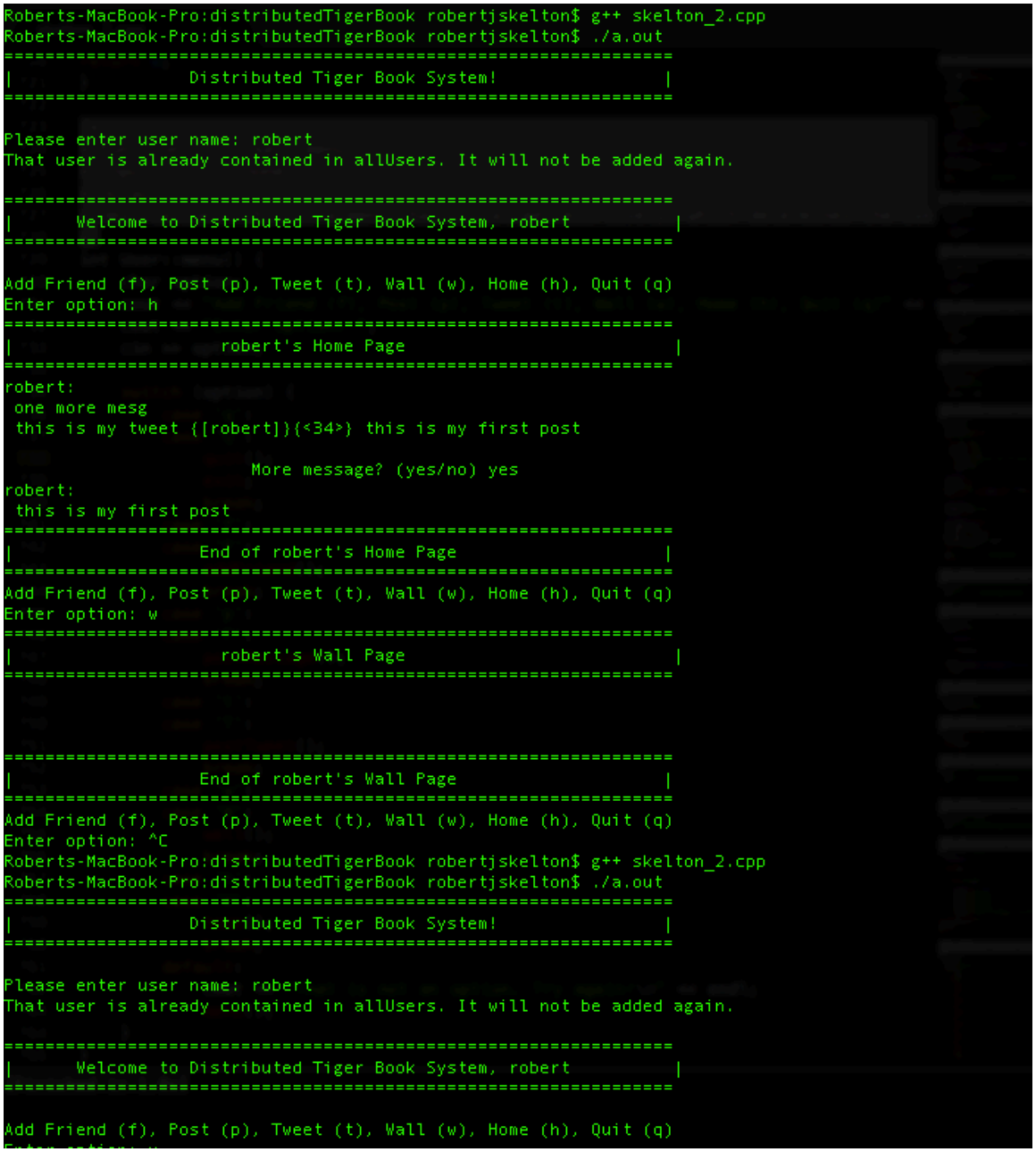
When entering user name, try: “Robert”, “{Robert”,”{[Robert”, “]Robert”, “]}Robert”, “{<Robert”, “>}Robert”. “{Robert”, “]}Robert”, “{<Robert”, and “>}Robert” should give an error, and prompt you to enter a new user name. The other entries should work fine as a user name. This works as expected.

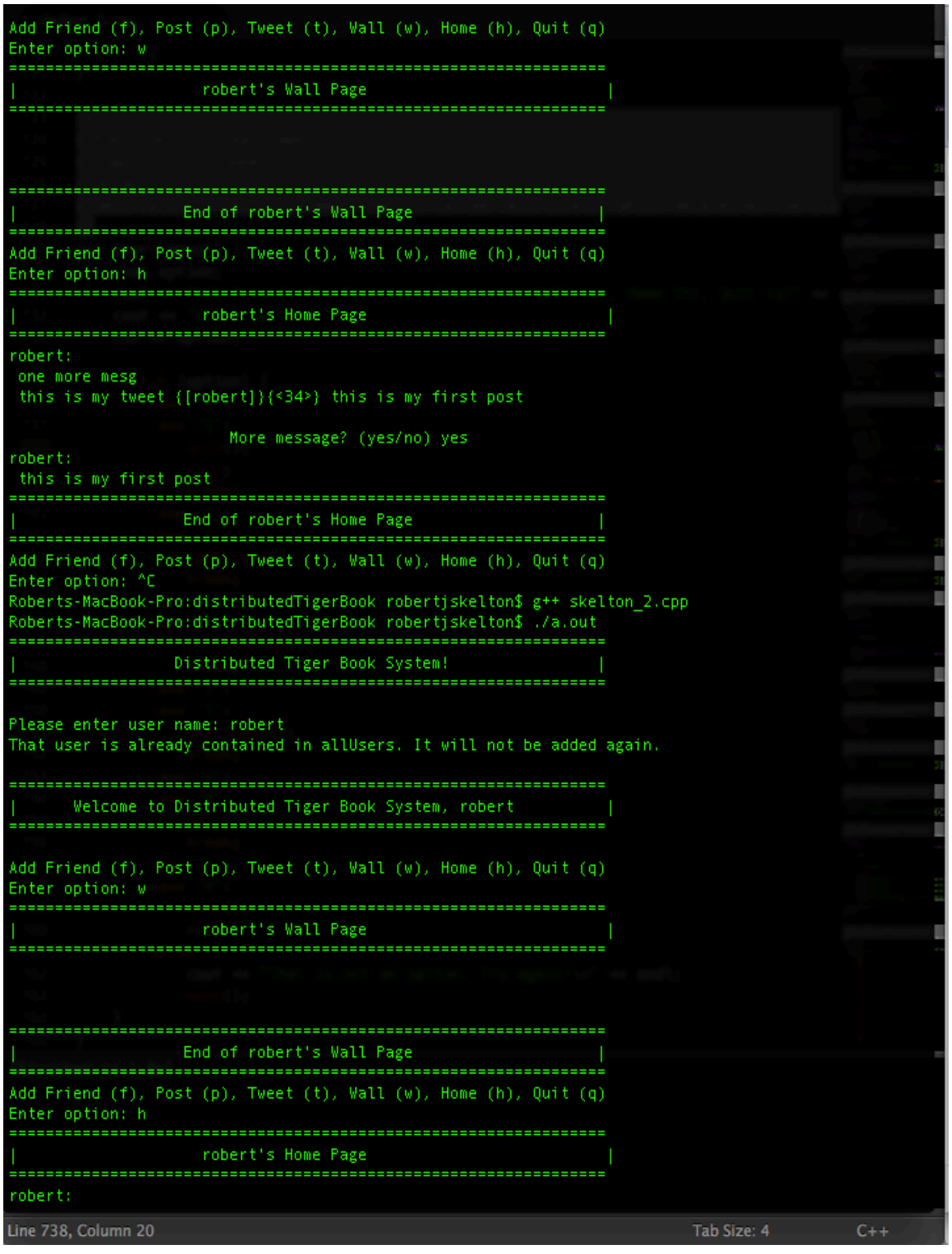


When selecting from the main menu, try: ‘f’, ‘p’, ‘t’, ‘w’, ‘h’, ‘q’, ‘F’, ‘P’, ‘T’, ‘W’, ‘H’, ‘Q’, ‘j’, ‘J’, ‘l’, ‘6’, and ’hello’.

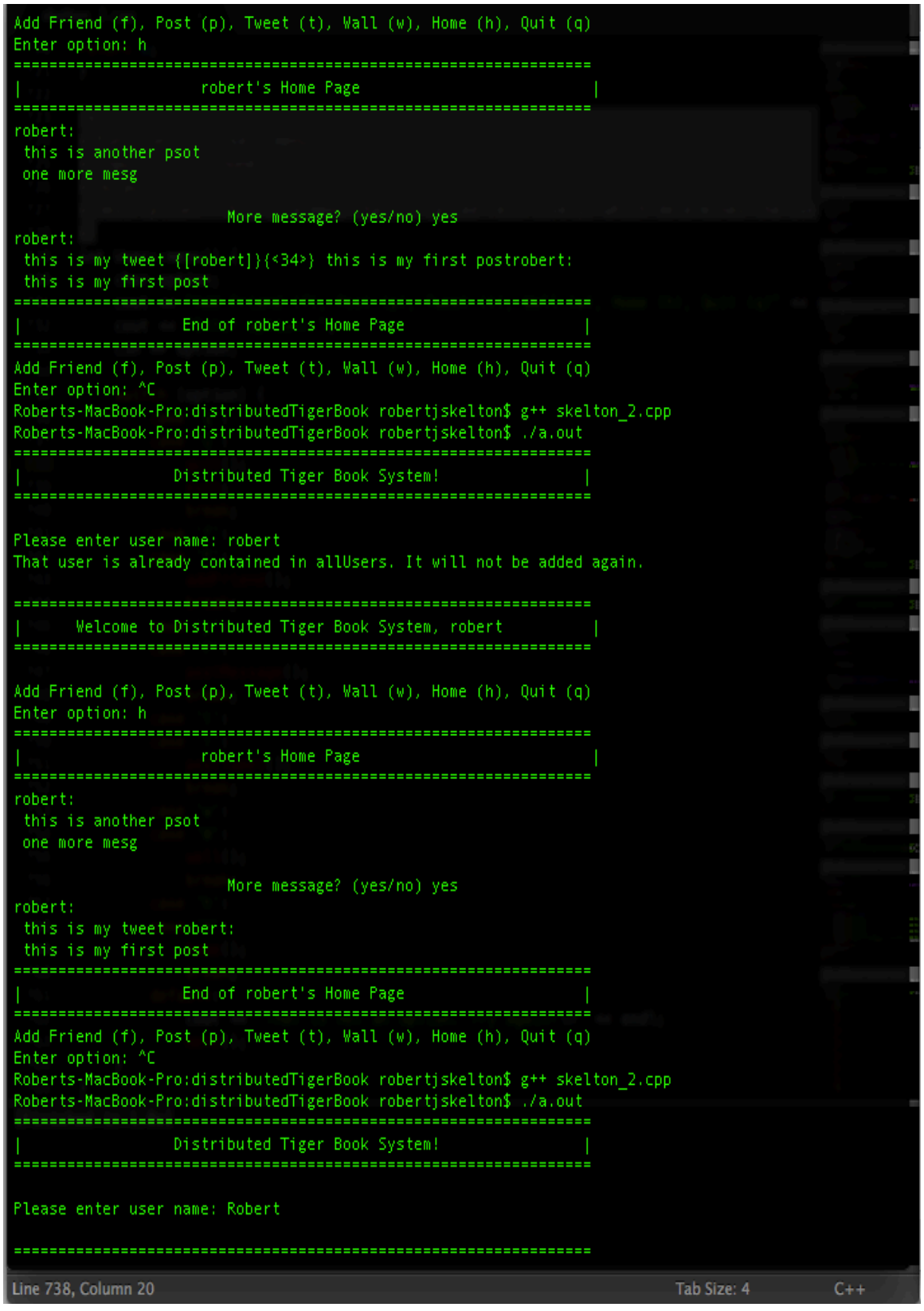
‘j’, ‘J’, ‘l’, ‘6’, and ’hello’ should give you an error and re prompt the main menu for you. The other entries are acceptable menu choices, and the program will continue on as normal. They all work as expected except for “hello”. “hello” will be parsed as “h”, so will call

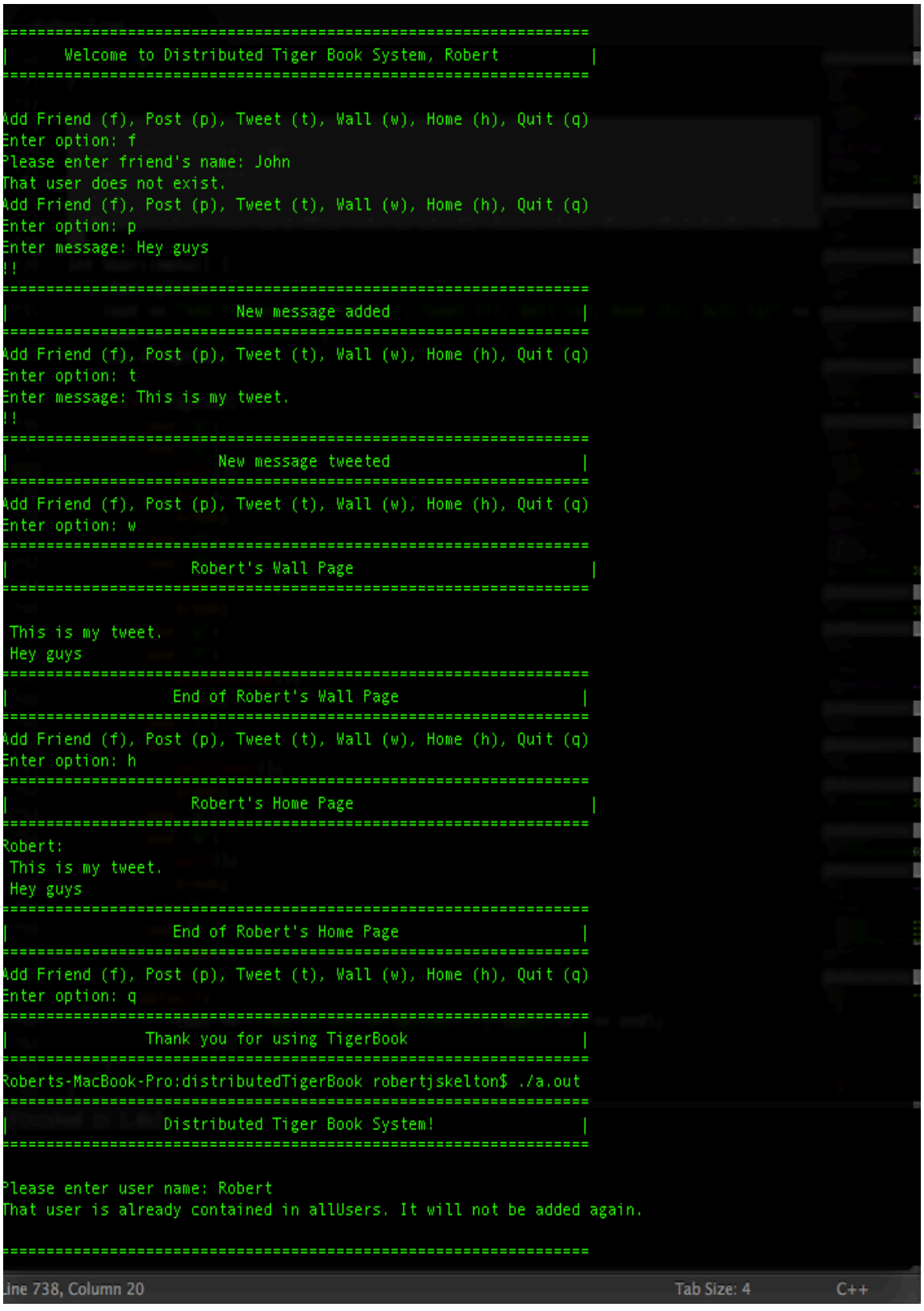
home. I decided to keep it this way because this is a minor bug, and should never be experienced by the user. “j”, “J”, “6”, and “l” will give you a message saying that those are not options, and prompt you to type in another option.

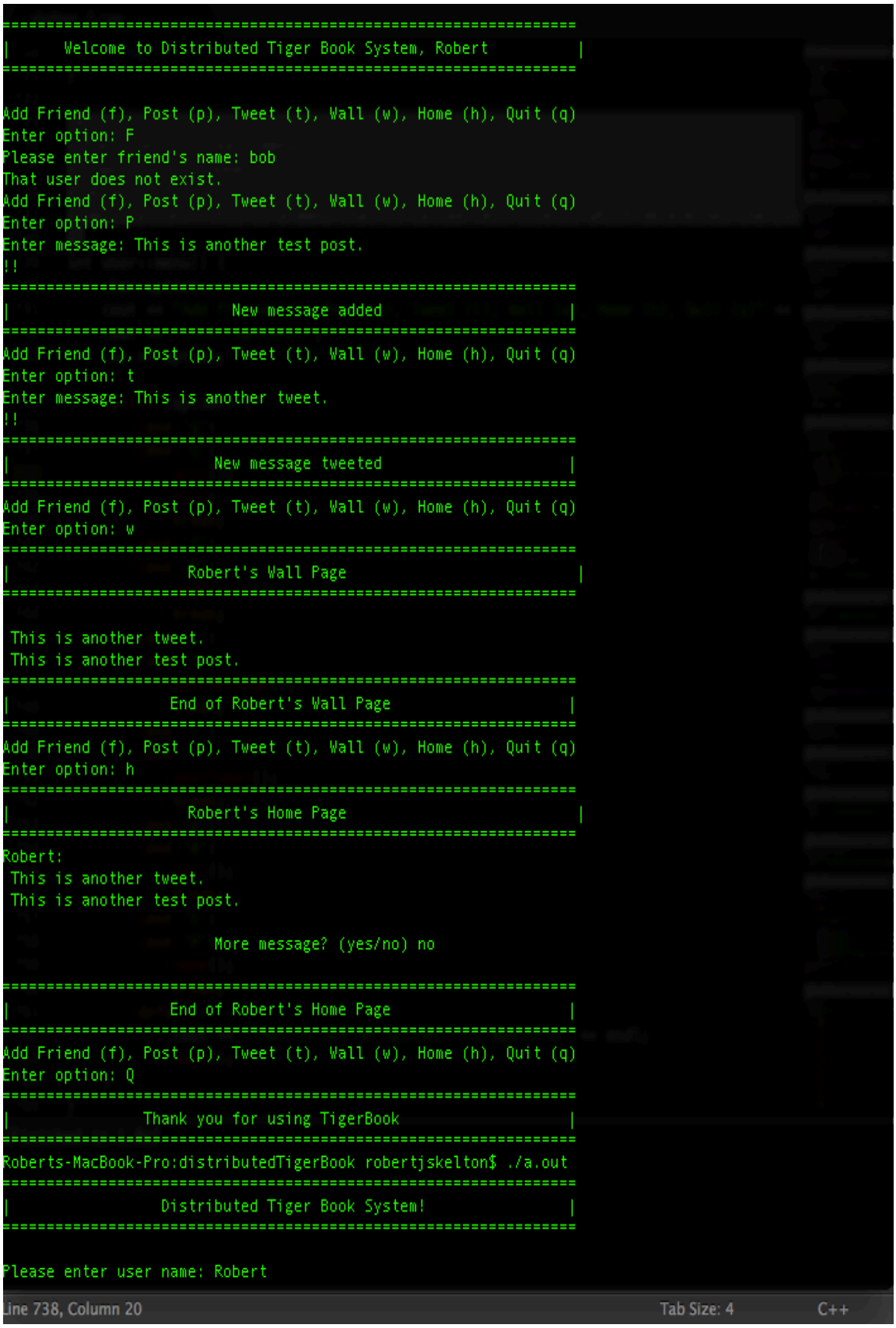


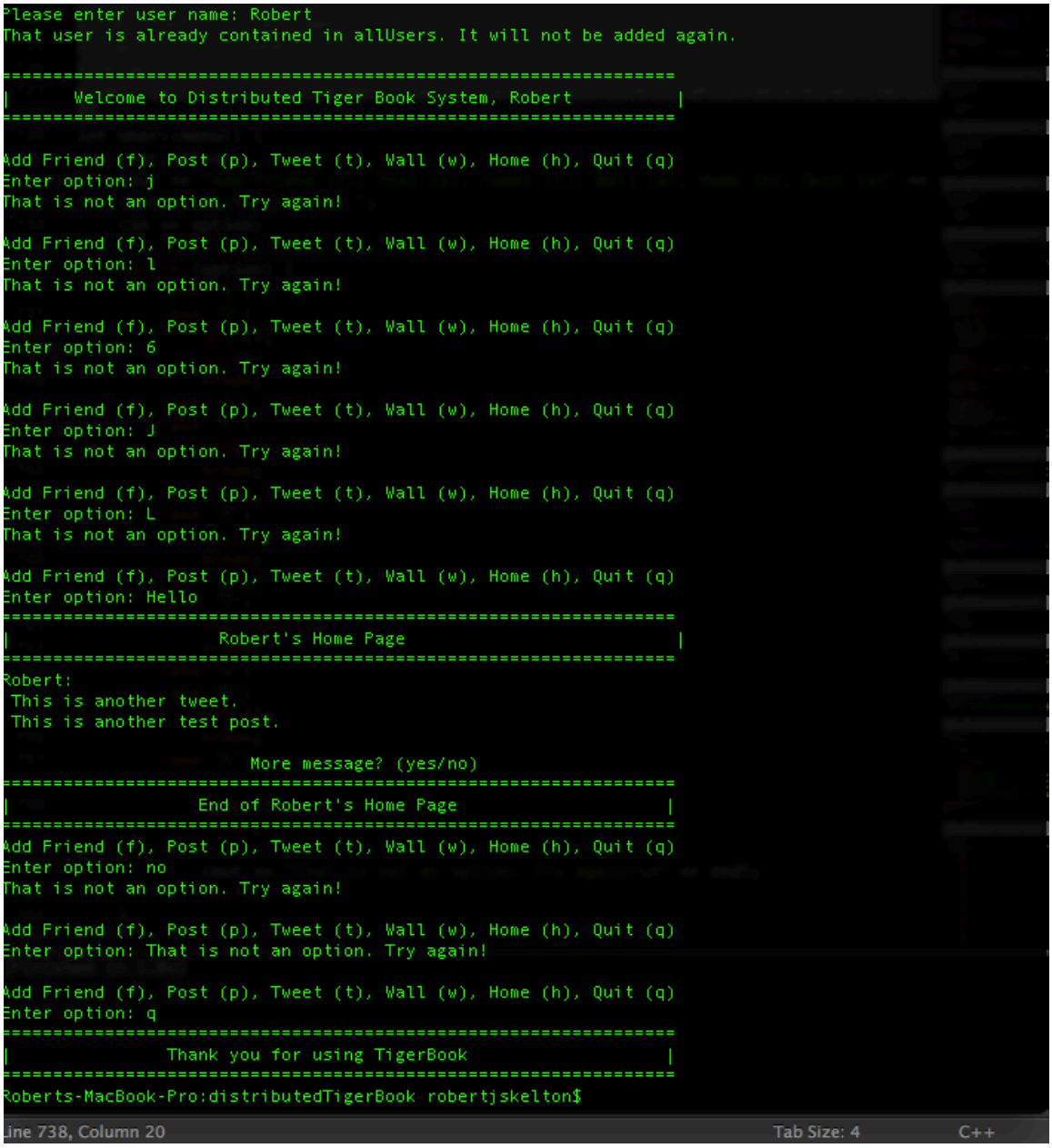

Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: w
robert's Wall Page
End of robert's Wall Page
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
robert's Home Page
robert:
one more mesg
this is my tweet {[robert]}(<34>) this is my first post
More message? (yes/no) yes
robert:
this is my first post
End of robert's Home Page
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: ^C
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ g++ skelton_2.cpp
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
Distributed Tiger Book System!
Please enter user name: robert
That user is already contained in allUsers. It will not be added again.
Welcome to Distributed Tiger Book System, robert
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: w
robert's Wall Page
End of robert's Wall Page
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
robert's Home Page
robert:
Line 738, Column 20
Tab Size: 4
C++

```
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
================================================================
|                  robert's Home Page                          |
================================================================
robert:
 one more mesg
 this is my tweet

                        More message? (yes/no) yes
robert:
 this is my first post
================================================================
|               End of robert's Home Page                      |
================================================================
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option:
p
Enter message: this is another psot
!!
================================================================
|                         New message added                    |
================================================================
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
================================================================
|                  robert's Home Page                          |
================================================================
robert:
 this is another psot
 one more mesg

                        More message? (yes/no) yes
robert:
 this is my tweet {[robert]}robert:
 this is my first post
================================================================
|                End of robert's Home Page                     |
================================================================
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: ^C
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ g++ skelton_2.cpp
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
================================================================
|               Distributed Tiger Book System!                 |
================================================================

Please enter user name: robert
That user is already contained in allUsers. It will not be added again.

================================================================
|      Welcome to Distributed Tiger Book System, robert        |
================================================================
```


Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
robert's Home Page
robert:
this is another psot
one more mesg
More message? (yes/no) yes
robert:
this is my tweet {[robert]}{<34>} this is my first postrobert:
this is my first post
End of robert's Home Page
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: ^C
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ g++ skelton_2.cpp
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
Distributed Tiger Book System!
Please enter user name: robert
That user is already contained in allUsers. It will not be added again.
Welcome to Distributed Tiger Book System, robert
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
robert's Home Page
robert:
this is another psot
one more mesg
More message? (yes/no) yes
robert:
this is my tweet robert:
this is my first post
End of robert's Home Page
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: ^C
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ g++ skelton_2.cpp
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
Distributed Tiger Book System!
Please enter user name: Robert
Line 738, Column 20
Tab Size: 4
C++

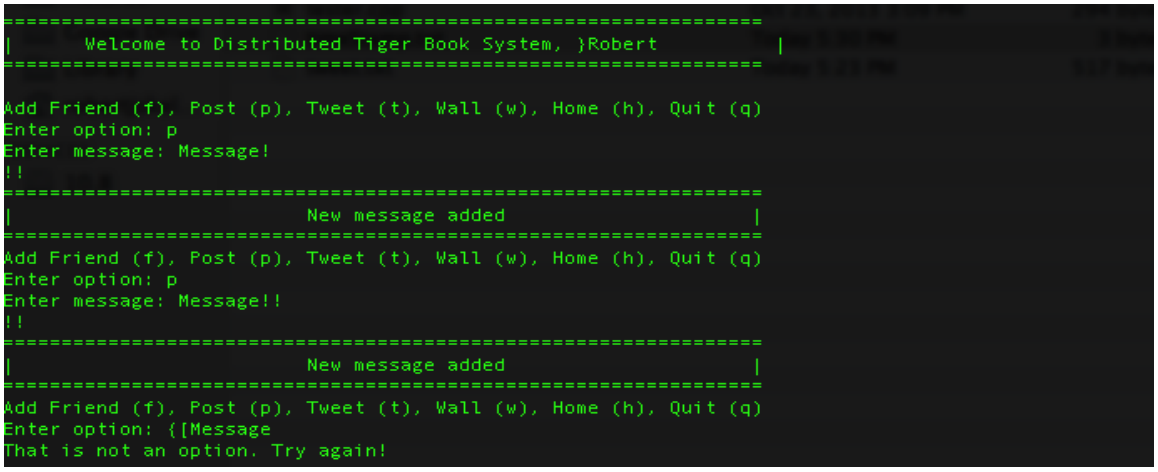

| Welcome to Distributed Tiger Book System, Robert |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: f
Please enter friend's name: John
That user does not exist.
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: p
Enter message: Hey guys
!!
| New message added |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: t
Enter message: This is my tweet.
!!
| New message tweeted |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: w
| Robert's Wall Page |
This is my tweet.
Hey guys
| End of Robert's Wall Page |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
| Robert's Home Page |
Robert:
This is my tweet.
Hey guys
| End of Robert's Home Page |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: q
| Thank you for using TigerBook |
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
| Distributed Tiger Book System! |
Please enter user name: Robert
That user is already contained in allUsers. It will not be added again.
Line 738, Column 20
Tab Size: 4
C++


| Welcome to Distributed Tiger Book System, Robert |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: F
Please enter friend's name: bob
That user does not exist.
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: P
Enter message: This is another test post.
!!
| New message added |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: t
Enter message: This is another tweet.
!!
| New message tweeted |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: w
| Robert's Wall Page |
This is another tweet.
This is another test post.
| End of Robert's Wall Page |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: h
| Robert's Home Page |
Robert:
This is another tweet.
This is another test post.
More message? (yes/no) no
| End of Robert's Home Page |
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: Q
| Thank you for using TigerBook |
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
| Distributed Tiger Book System! |
Please enter user name: Robert
Line 738, Column 20
Tab Size: 4
C++

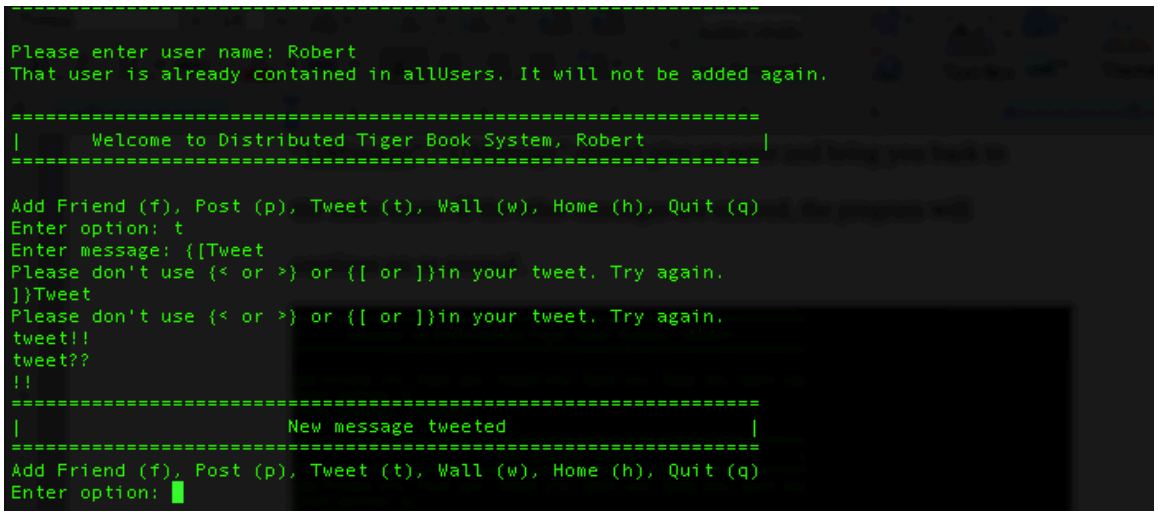


When entering a message, enter: “Message!”, “Message!!”, “!!”, “{[Message”, “]}Message”, “{<Message”, “]Message”, “[]Message”. “!!” should end the message entering process. “{<Message”, “]}Message”, “{[Message” should give an error and bring you back to the main menu. If the other messages are entered, the program will

continue on as normal.



When entering a tweet, enter: “Message!”, “Message!!”, “!!”, “{[Message”, “]}Message”, “{<Message”, “]Message”, “[]Message”. “!!” should end the message entering process. “{<Message”, “]}Message”, “{[Message” should give errors and bring you back to the main menu. If the other tweets are entered, the program will continue on as normal.



If you post “1”, “2”, “3”, ”4”: When showing your wall page, it

would first show a banner saying who's wall page you are viewing, and then should show:

4

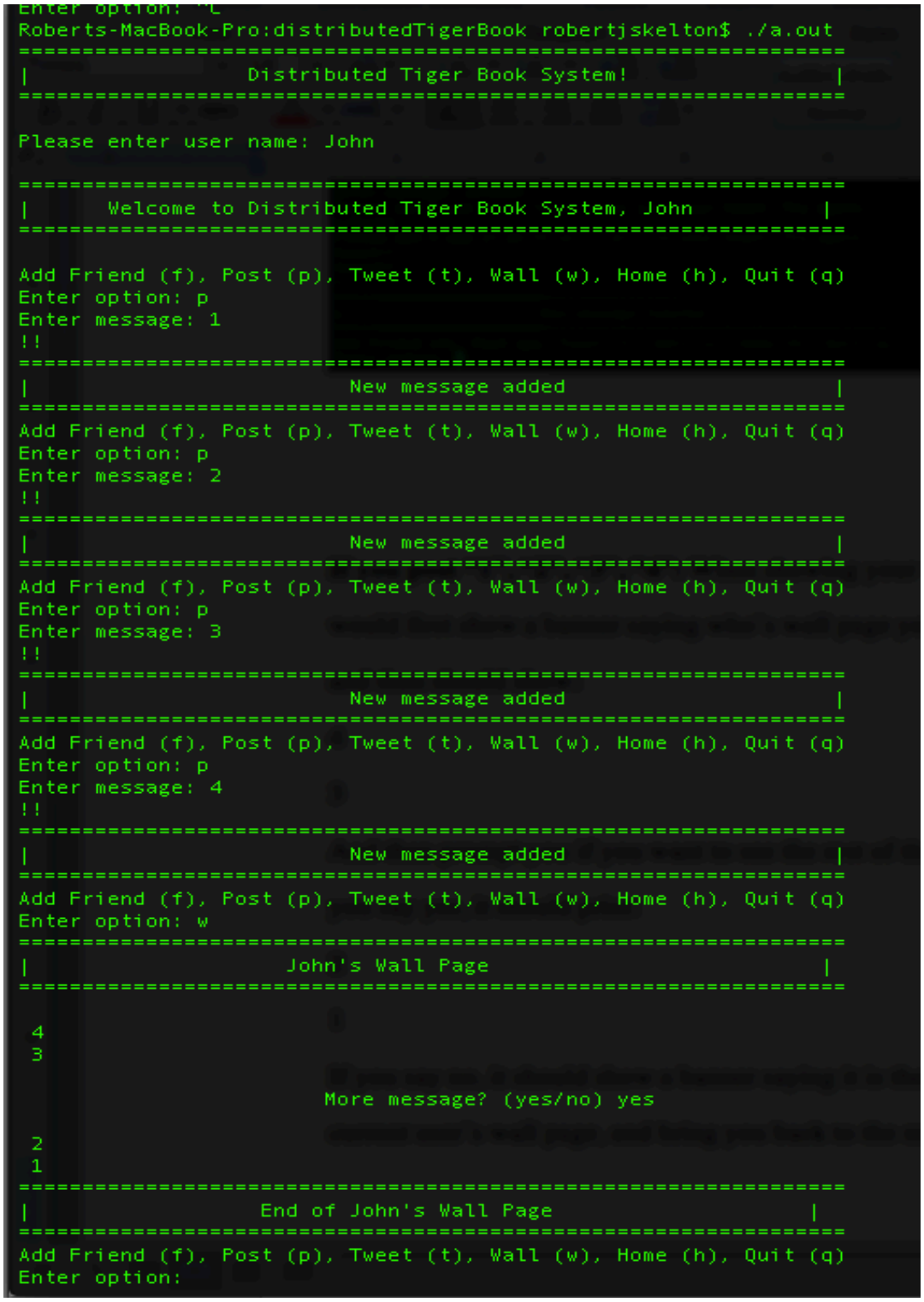
3

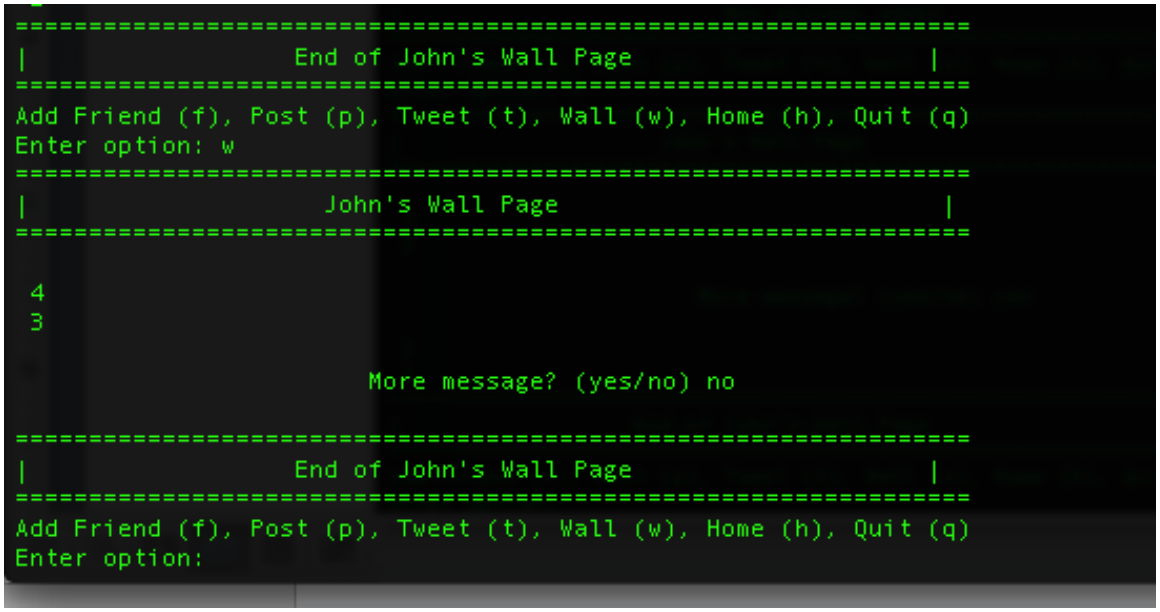
And then prompt you if you want to see the rest of the messages. If you say yes, it should print:

2

1

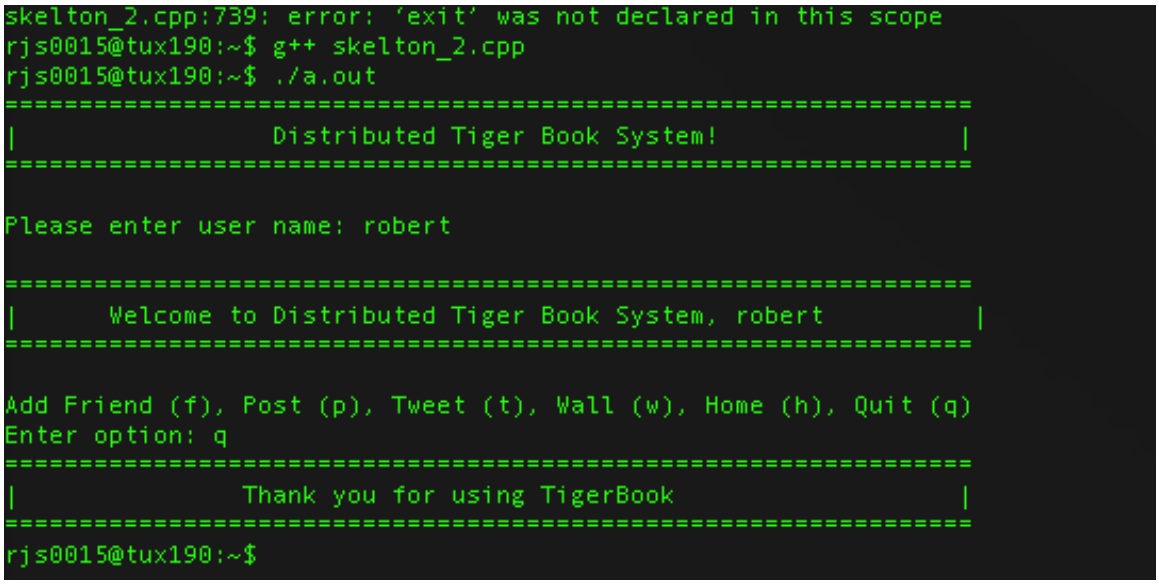
If you say no, it should show a banner saying it is the end of the current user's wall page, and bring you back to the main menu. All new posts will be on a new line, and the user's name will not be shown.


Enter option: ^C
Roberts-MacBook-Pro:distributedTigerBook robertjskelton$ ./a.out
Distributed Tiger Book System!
Please enter user name: John
Welcome to Distributed Tiger Book System, John
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: p
Enter message: 1
!!
New message added
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: p
Enter message: 2
!!
New message added
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: p
Enter message: 3
!!
New message added
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: p
Enter message: 4
!!
New message added
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option: w
John's Wall Page
4
3
More message? (yes/no) yes
2
1
End of John's Wall Page
Add Friend (f), Post (p), Tweet (t), Wall (w), Home (h), Quit (q)
Enter option:



Conclusion

This was a difficult project, but I learned more about Object-Oriented programming from it. I hope I earned better than an F this time. Enjoy!



Testing on Linux made me realize the G++ compiler on Linux did not like

the exit; command, which was a backup in my quit() function. This was as easy fix, as I just commented out that line of code. Other than that, it worked the same on OS X 10.8 and CentOS. Enjoy!

-Robert Skelton