Assignment 4: Bulletin Board

Collaboration Policy

You may not use code from any source (another student, a book, online, etc.) within your solution to this assignment. In fact, you may not even look at another student's solution or partial solution to this assignment. You also may not allow another student to look at any part of your solution to this exercise. You should get help on this assignment by coming to the instructor's or TA's office hours or by posting questions on Piazza (you still must not post assignment code publically on Piazza.) See the full Course Collaboration Policy here: Collaboration Policy

---

Problem Definition

In this assignment you will create an extremely rudimentary bulletin board with a simple command-line interface (like in the old days!).

This will require three collaborating classes - BBoard, User and Message. For the moment, the bulletin board will exist only inside the test harness (i.e. we won't store it to the file system yet), so you will construct a single BBoard object (giving it an appropriate title - e.g. "Jack's Amazing Bulletin Board"), and then invoke its methods.

First, you will "set it up" with a set of authorized Users (read in from a file);

then you will ask the user to login with a username and password, which you will check against the list you just read in;

and then you will "run" the board: the program will enter a sentinel-controlled loop asking the user to choose from a menu of actions (corresponding, obviously, to BBoard's other public methods)

- list all current posts
- create a new post
- quit the system

File Naming Specifications

The files that you submit should be named **main.cpp**, **BBoard.h**, **BBoard.cpp**, **Message.h**, **Message.cpp**, **User.h**, and **User.cpp**.
The names are case-sensitive.

Class Specifications

BBoard Class interface

Changing the public interface will result in a 0 on the assignment.

Encapsulated data (private member variables)

- `title` - string : *title of the board; initialized via constructor*
- `user_list` - vector<User> : *list of members; initially empty, then populated via setup()*
- `current_user` - User : *user who is currently logged in; initialized by c'tor to the default User, then set via login()*
- `message_list` - vector<Message> : *list of messages; initially empty*

Public Member Functions

This is the public interface of the class - the contract with the user.

BBoard( )
default constructor that creates a board with a default title, empty user & message lists, and the "default" User

BBoard(const string &ttl)
same as default c'tor, except that it sets the title of the board.

void setup(const string &input_file)
imports all the authorized users from an input file, storing them as User objects in the vector user_list
The name of the input file is passed in as a parameter to this function.
See below for file details.

void login( )
asks for and validates current user/password
This function asks for a username and password, and checks the user_list vector for a matching User.

This function must take in both username and password (even if the username doesn't exist, ie password is always requested), unless username is “q” or “Q”, in which case quit.

If a match is found, it sets current_user to the identified User from the list.
If not, it will keep asking until a match is found or the user types: "q" or "Q" as the username.

You can assume we do not have a member with name "q" or "Q". When the user chooses to quit, say "Bye!" and call "exit(0)" (see cstdlib reference) command to exit the program.
Read output specifications for details.

void run( )

This function contains the main loop of the BBoard.

Note: if login() has not set a valid current_user, this function must abort without showing the menu - i.e. there must be no way to get around logging in!

Question: how will you know if a valid user is not logged in?

In the main loop, you should display the following menu items:
"Display Messages ('D' or 'd'), "Add New Message('N' or 'n')", and "Quit ('Q' or 'q')".

If the user selects one of these menu items, the corresponding method of BBoard should be invoked; with any other input, the user should be asked to try again.

'Q'/'q' should terminate the program by invoking exit(0), as described for login.
See suggested private functions for details.

Suggested Private Member Functions

These are "helper functions": member functions that will never be needed by a user of the class - they are needed by the public interface functions. You are free to change/add to/remove these functions as you wish:

void add_user(istream &infile, const string &name, const string &pass)

This function can be called from the setup function to add the users from file to user_list.

bool user_exists(const string &name, const string &pass) const

Checks if the username/password combination matches any of the Users in the user_list.

Returns true if a match is found, false otherwise. Useful for login function.

void display() const

This function would be used by the interface function run. It displays all messages in message_list.
The output must be formatted ***exactly*** as in the example below.

void add_message()

This function would be used by the interface function run. It asks the user to input a message. Every message includes a subject line and a message line (both are single lines, so you can ignore newlines). e.g.,

```
Subject: "Thanks"
Body: "I would like to thank you for this awesome board."
```

Hint: You can use getline function to get the whole line: see getline reference.
Of course, the username of current_user must also be recorded in the Message object.

User and Message Classes

The User and Message classes are two simple classes used by the BBoard class only - they are ***not*** used (or included) in main.cpp, only in BBoard.h and BBoard.cpp
Note that - by design - the User class will ***never*** "report" a User's password - you can only ask it to check if a given username/password matches. This is one place where encapsulation is the only way to go!
Changing the public interface will result in a 0 for the assignment.

User.h

```
//inclusion guards

//includes

class User
{
  private:
    string username;
    string password;

  public:
    //creates a user with empty name and password.
    User();

    // creates a user with given username and password.
    User(const string& uname, const string& pass);
```

```
    //returns the username
    string get_username() const;

    // returns true if the stored username/password matches with the
    // parameters. Otherwise returns false.
    // Note that, even though a User with empty name and password is
    // actually a valid User object (it is the default User), this
    // function must still return false if given a empty uname string.
    bool check(const string &uname, const string &pass) const;

    // sets a new password. This function will not be used in the
    // current assignment.
    void set_password(const string &newpass);
};

//end inc guards
```

Message.h

```
//inclusion guards
//includes

class Message
{
  private:
    string author;
    string subject;
    string body;

  public:
    //default constructor
    Message();

    //Constructor with parameters
    Message(const string &athr,
            const string &sbjct,
            const string &body);

    //displays the message in the given format. See output specs.
    //Note: R'Sub will require an endl as the last output of this
    //function.
    void display() const;
};
//end inc guards
```

Main Function

In the main function, you should create a BBoard object e.g., `BBoard bb( user_bb_title );`, giving the title any name you want, and then import the list of authorized users with setup(name of input file). The name of the input file you pass into the setup function will be a **command line argument**. This file will be in the same directory as the executable.

Next, use the login function to display a welcome message, and then ask the user to login (provide username and password); the function then sets the current_user (or quits).

Finally you'll just call the run function which includes the main loop.

Input Specifications

User File Specifications:

The name of the file is passed in to the main function as a command line argument (like this):

```
./a.out users1.txt
```

The user file is formatted as follows:

```
user1 pass1
user2 pass2
user3 pass3
...
end
```

This file will be read in (via the setup function) at the very start of the program, to establish the list of authorized users.

As you can see, the number of users is not predetermined - the list continues until the sentinel "end" is encountered on a line by itself. Also, you must terminate your text file with a newline.
Here is an example file that you may use as a starting point:

users1.txt

```
ali87   8422
ricq7   bjk1903
messi   buneyinnessi
mike    ini99ou
jenny   Y00L11A09
end
```

Keyboard Input Specifications

- When the user is asked to select a menu item, the only valid inputs are
  'D', 'd', 'N', 'n', 'Q' and 'q' - not "quit","new", "Quit" etc.
- Every message body and subject consists of a single line. (If you use 'cin >>...' it will not work, since this command does not distinguish newline from other whitepaces
  (see section 9.7 in your zyBook for more information on this).
- See sample output for details.
- On unrecognized input, the program should terminate.

Output Specifications:

Read the examples below and make sure the output matches **exactly** with the sample runs below (given the same inputs).

Sample run 1

```
$./a.out users1.txt


Welcome to Jack's Amazing Bulletin Board
Enter your username ('Q' or 'q' to quit): muzo
Enter your password: cs12
Invalid Username or Password!

Enter your username('Q' or 'q' to quit): ricq7
Enter your password: mmee00
Invalid Username or Password!

Enter your username('Q' or 'q' to quit): Q
Bye!
```

Sample run 2

```
$./a.out users1.txt



Welcome to Jack's Amazing Bulletin Board
Enter your username ('Q' or 'q' to quit): messi
Enter your password: buneyinnessi
Welcome back messi!

Menu
  - Display Messages ('D' or 'd')
```

```
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: D

Nothing to Display.

Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: N

Enter Subject: Thanks
Enter Body: Thank you for this amazing board.
Message Recorded!

Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: N

Enter Subject: Totenham vs. Arsenal
Enter Body: Have you watched the game on wednesday? It was amazing far
better than El Clasico.
Message Recorded!

Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: N

Enter Subject: Output Specifications
Enter Body: So, we just have to make the program run exactly like these
samples.
Message Recorded!

Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: D

------------------------------------------------
Message #1: Thanks
from messi: Thank you for this amazing board.
------------------------------------------------
```

```
Message #2: Totenham vs. Arsenal
from messi: Have you watched the game on wednesday? It was amazing far
better than El Clasico.
-----------------------------------------------
Message #3: Output Specifications
from messi: So, we just have to make the program run exactly like these
samples.
-----------------------------------------------

 Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: N

Enter Subject: Easy?
Enter Body: This assignment does not look so easy, any extensions?
Message Recorded!

Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: D

-----------------------------------------------
Message #1: Thanks
from messi: Thank you for this amazing board.
-----------------------------------------------
Message #2: Totenham vs. Arsenal
from messi: Have you watched the game on wednesday? It was amazing far
better than El Clasico.
-----------------------------------------------
Message #3: Output Specifications
from messi: So, we just have to make the program run exactly like these
samples.
-----------------------------------------------
Message #4: Easy?
from messi: This assignment does not look so easy, any extensions?
-----------------------------------------------

Menu
  - Display Messages ('D' or 'd')
  - Add New Message ('N' or 'n')
  - Quit ('Q' or 'q')
Choose an action: Q
Bye!
```

Compiling and Testing

You can compile the entire program, generating an executable named a.out, with the following command line:

```
g++ -W -Wall -Werror -ansi -pedantic main.cpp Message.cpp
BBoard.cpp User.cpp
```

But you should compile the classes separately, without generating an executable, with a command line similar to:

```
g++ -W -Wall -Werror -ansi -pedantic -c Message.cpp
```

replacing Message.cpp with whichever class you are trying to compile of course. And then to compile the result

```
g++ -W -Wall -Werror -ansi -pedantic main.cpp Message.o BBoard.o
User.o
```

What to Submit

For this assignment you will turn in the following files to R'Sub (galah.cs.ucr.edu):

- main.cpp (Your test harness.)
- BBoard.h (with exact interface given in specs.)
- BBoard.cpp
- User.h (with exact interface given in specs.)
- User.cpp
- Message.h (with exact interface given in specs.)
- Message.cpp