

**Virginia Tech**  
**Bradley Department of Electrical and Computer Engineering**  
**ECE-3574: Applied Software Engineering Fall 2014**

**Homework 8**

**Submission Details**

You must submit the solutions for this homework as an electronic submissions using Scholar (under ECE3574 → Assignments → Homework 8). The submission must be a gzipped tar file (.tar.gz) with your source code. Include all necessary project files, but no binary or compiled files. Your program will be run to evaluate its correctness, and the source code will be reviewed for adherence to the Qt programming style. Your program must run on Ubuntu 14.04.1 and compile/build using the GNU C/C++ compiler and the qmake/make tools. The following information must be included at the top of each of your source files as comments: your full name, your student ID number, your email address, class (ECE 3574), and the title of the assignment (Homework 8). The submitted file must be given a name in the following form: ***LAST\_FIRST\_hw8.tar.gz*** where LAST is your last or family name and FIRST is your first or given name. Paper, email or Drop Box submissions will not be accepted. All work must be submitted by the announced due date/time. **Late submissions will not be accepted!** (Don't do it! You have been warned!)

**Questions**

Use the Homework 8 forum in the Discussion Board area of the class web site to ask questions about this assignment. Do not post questions that contain specific information about the solution.

**Honor Code**

As stated in the syllabus, in working on homework and projects, discussion and cooperative learning are allowed. However, copying or otherwise using another person's detailed solutions to assigned problems is an honor code violation. See syllabus for details.

## Learning Objectives

The primary learning objectives of this assignment include the use of event driven programming abstractions, graphical widgets construction of graphical applications, the use of efficient high-level programming techniques using libraries, generics containers and the use of design patterns such as the composite pattern and serializer patterns. Additional learning objectives include the use of classes, inheritance and polymorphism.

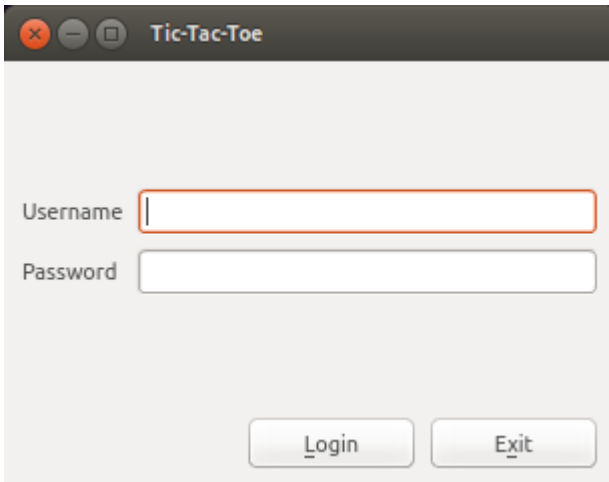
## Homework

Write a “login” program that registers users, saves the user-name and password in a data file and logs them into the application. The users should be able to change their passwords. The application is a Tic-Tac-Toe game.

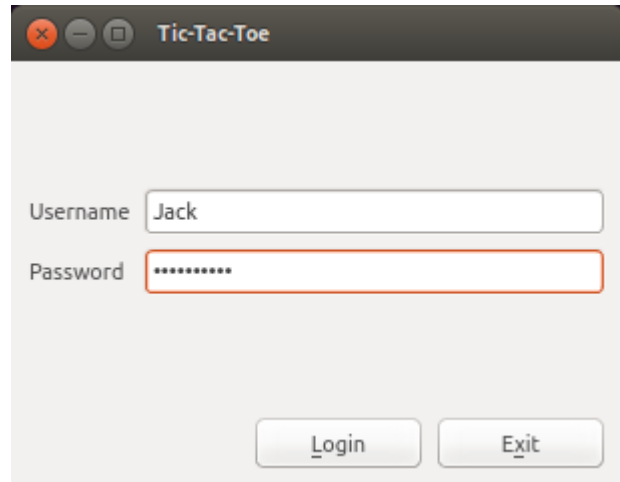
## Specifications

1. The GUI may be created with QtDesigner. Read the notes at the end of the document for more details.
2. The application will have 5 forms. Each form will be in its own QWidget-derived class.
  - a. **Main Login panel** – This panel lets the registered user log-in to the application
  - b. **Register User panel** – This panel registers a new user.
  - c. **Welcome User panel** – Once the login is successfully done, this would be the welcome screen for the user.
  - d. **Change Password panel** – This panel changes the password of the logged-in user
  - e. **Tic-Tac-Toe game panel** – This panel starts a new tic-tac-toe game.
3. **NOTE:** There is only one main window and all the panels listed above must be shown inside the main window, one at a time, based on the selection.
4. The main window has a tool-bar with three pull down menus (“User”, “Game” and “Edit”).
  - a. The “User” menu has the following actions
    - i. Register User – This action shows the “Register User Panel”
    - ii. Logout User – This action logs out the current user and shows the “Main Login Panel”
    - iii. Exit – This action exits the application
  - b. The “Game” menu has the following actions
    - i. New game – This action starts the new game and shows the “Tic-Tac-Toe” game panel
    - ii. End game – This action ends the game and show the “Welcome User Panel”
  - c. The “Edit” menu has the following actions
    - i. Change Password – This action shows the “Change Password Panel”

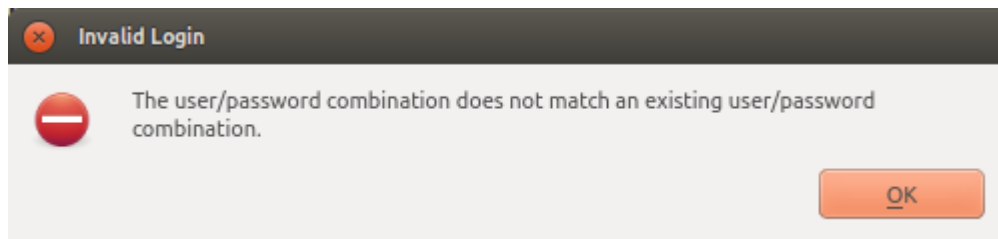
## Main Login Panel



*Login Panel*



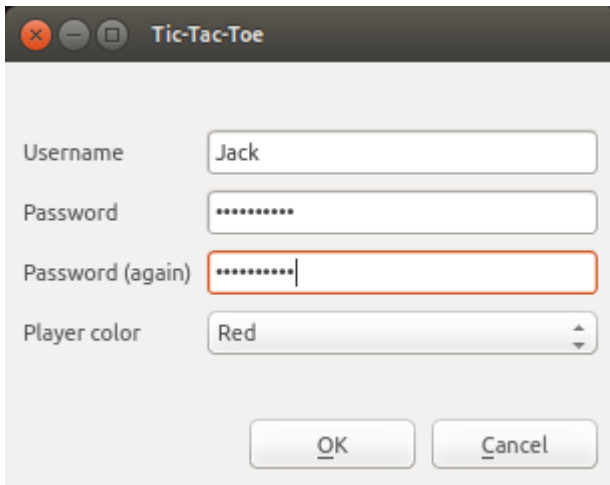
*Login with username/password*



*Login error for unregistered users and registered user with incorrect passwords*

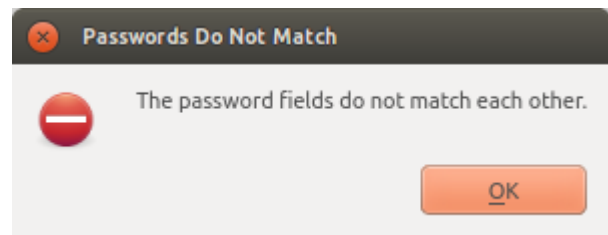
1. When the “Logout” action is clicked, shows the Login panel.
2. The application starts in Main Login panel.
  - a. In the Main Login panel, the user is asked for the user name and password.
  - b. Registered users with matching passwords must be logged in
  - c. Unregistered users and registered users with invalid passwords must be denied access
  - d. To register, the users can choose the “Register User” action from the menu “User”
3. Once the user has been registered, in the “Main Login panel” when the user tries to login using the correct user-name and password combination then clicking on the “Login” button must successfully log the person in and show the “Welcome Panel
4. Clicking login with an unregistered username must show a popup saying that the username/password combination is invalid
5. Clicking login with a password that doesn’t match the username must show a popup saying that the username/password combination is invalid. (Note that this is the same message as above so as to better guard which usernames are registered).
6. If the user clicks on the “Exit” button the program exits.

## Register User Panel

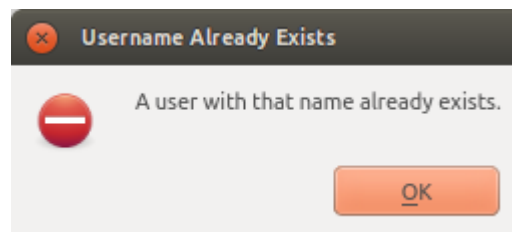


A screenshot of a 'Tic-Tac-Toe' application window. It contains a 'Register User Panel' with the following fields: 'Username' (containing 'Jack'), 'Password' (masked with dots), 'Password (again)' (masked with dots and highlighted with a red border), and 'Player color' (a dropdown menu showing 'Red'). At the bottom are 'OK' and 'Cancel' buttons.

*Register User Panel*



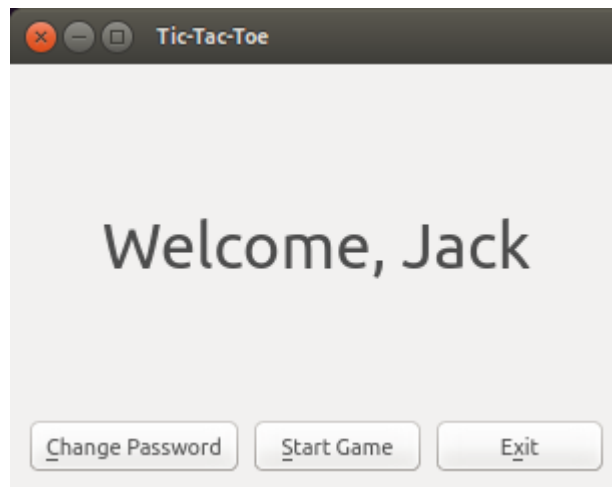
*New Password Mismatch*



*User Already Exists*

1. When the “Register User” action is clicked, it must show the Register user panel.
2. In this Register User panel, the user is asked for the following details. (These must be fields)
  - a. Desired user name
  - b. Password
  - c. Re-enter password
  - d. A drop down menu with names of colors. The user must select one color out of red, green or blue. This color will be used to determine their tic-tac-toe symbol color.
2. When the user click “Ok”, the program must check for the following exceptions
  - a. If the user already exists in the “passwords.dat” file, the program must show a popup with the message “User name already exists, please pick another”.
  - b. If the “password” and “re-enter Password” do not match, the program must show a popup with the message “Passwords do not match, please re-enter”
3. After clicking “Ok”, if all the conditions are met, the entry must be stored in the “passwords.dat” file and the program must go back to the “Main Login panel”
4. If the user clicks on “Cancel” the program must not store anything and go back to “Main Login Panel”

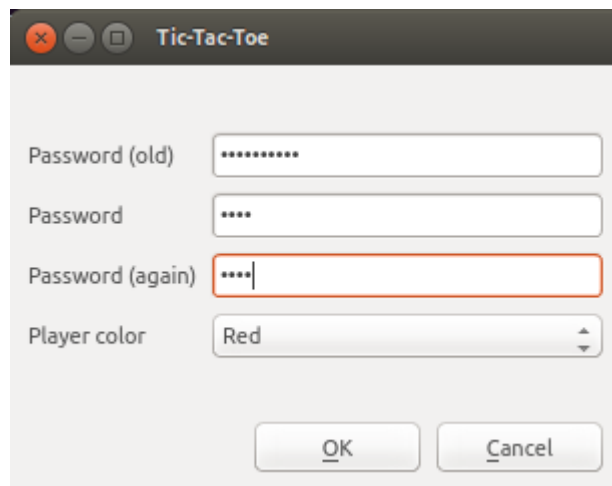
### Welcome User Panel



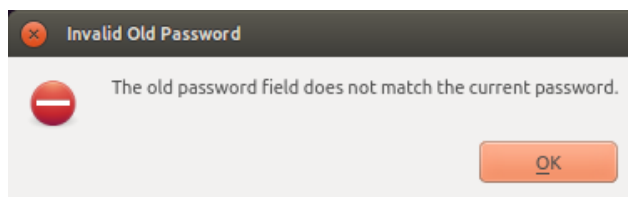
*Welcome User Panel*

1. The welcome panel must show a welcome message “Welcome, <username>” and it must have three buttons:
  - a. Change Password – This button must show the “Change password panel”
  - b. Start Game – This button must start a new game and show the “Tic-Tac-Toe game panel”
  - c. Exit – This button must exit the application

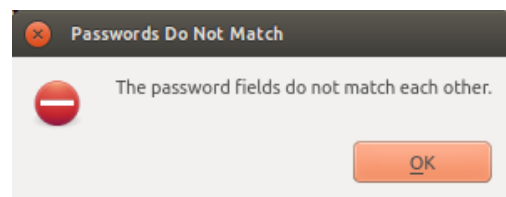
### Change Password Panel



*Change Password Panel*



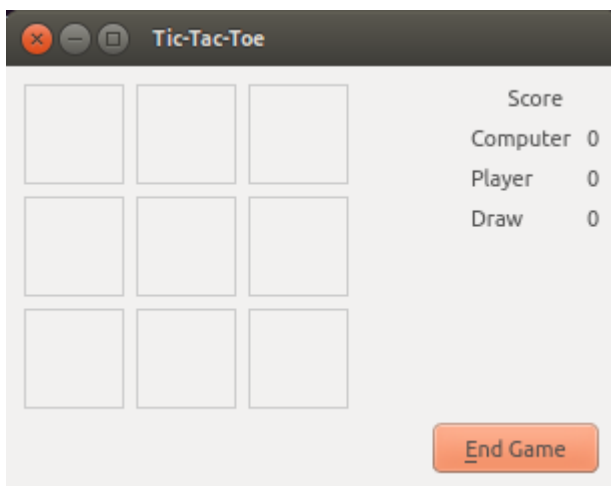
*Incorrect Old Password*



*New Password Mismatch*

1. When the “Change Password” action is clicked it must show the “Change Password Panel”.
2. The following fields must be shown on this panel
  - a. Old password of the user
  - b. New password
  - c. Re-enter new password
  - d. Drop down menu to select color
2. On clicking “Ok”, the following error conditions must be checked:
  - a. If the old password given by the user and the actual old password do not match, show a pop-up message “Old password does not match, please re-enter”
  - b. If the “new password” and “re-enter new password” fields do not match, show a pop-up message “The passwords do not match, please re-enter”.
3. If all the entries are correct, the program must change the password for that user as well as update the player’s color and go back to the “Welcome User Panel”. This change must be written back to the passwords.dat file.
4. If the user clicks on “Cancel”, the password must not be changed and the program must go back to the “Welcome User Panel”

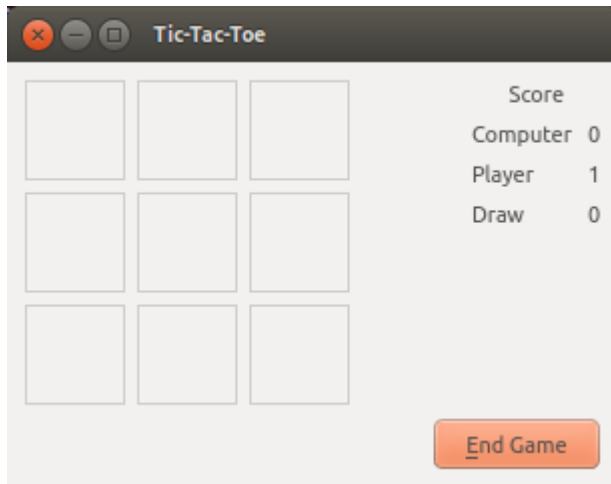
### Tic-Tac-Toe Game panel



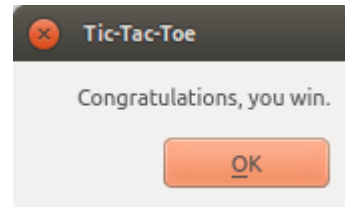
*Game Panel*



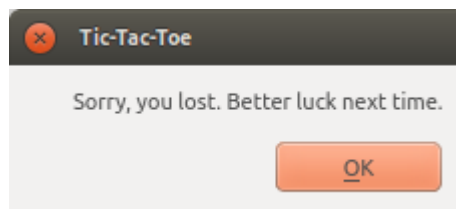
*Player/Computer Symbols*



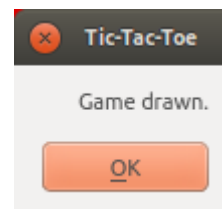
*Game Panel with 1 Win*



*Win Popup*



*Loss Popup*



*Draw Popup*

1. This panel must show a 3x3 grid for the tic-tac-toe game.
2. The user selects which cell of the grid to place their symbol with a mouse click in that cell.
3. The grid color and style is up to you but the color of the “X” and “O” markers must match the color the logged in user chose during registration or the when they changed their password, whichever is most recent.
4. General rules for the tic-tac-toe apply here – the person getting three in a row (diagonally, horizontally or vertically) wins the game.
5. The program must play against the user as “Computer”
6. The “user” must always make the first move and must always be given the “X”
7. The “Computer” must play second and must use “O”
8. If the game ends in draw, a popup must be shown saying “game drawn”. Clicking “Ok” on the pop-up message causes the game to re-start (The user must make the first move)
9. If the user wins, there must be a pop-up message “Congratulations, you win”. cOn clicking “Ok” on the pop-up message causes the game to re-start (The user must make the first move)
10. If the Computer wins, there must be a pop-up message “Sorry, you lost. Better luck next time”. Clicking “Ok” on the pop-up message causes the game to re-start (The user must make the first move).
11. If the user clicks on the action “End Game” under Game menu or clicks the “End game” button on the panel, the game must end and the program must show the “Welcome User Panel”.
12. There must be a score card on the panel which must show the count of the number of games won by the “Computer”, the “User” and the number of game that were drawn. These counters are reset when the user leaves the game panel.

### Passwords File

1. The program must save the information in a **binary** file called “passwords.dat”. The users’ passwords must be protected by QCryptographicHash’s implementation of the SHA1 algorithm. The output of a cryptographic hash is called a digest. For more information about using the cryptographic hash refer to the notes at the end of the assignment. When data is saved to the file it must follow a specific format.
2. Before reading or writing to the file you must use QDataStream::setVersion with the value QDataStream::Qt\_4\_6 on the data stream interfacing to the file regardless of your installed version of Qt.
3. All password records must be serialized by QHash. If you put all your records into a QHash you must be able to save and read everything in the hash table using one line each.  

```
dstream << listing; // writing
dstream >> listing; // reading
```

Where dstream is the QDataStream handling input and output from your file and listing is a QHash<QString, T> where T is your record data type. Note that the appropriate streaming operators must be defined on your record data type for this to work.

```
QDataStream& operator<<(QDataStream& stream, const T& val);
QDataStream& operator>>(QDataStream& stream,      T& val);
```
4. In the QHash the record will be associated with a username but the record itself will not contain the username. When serialized, QHash will output the username as a QString key along with the record (type T) as the value.
5. All records must be written out and read in, in the following order and in the following format
  - a. Color: as a QString
  - b. Password: convert it from a QString to a QByteArray using QString::toUtf8() and pass it through the SHA1 algorithm once which results in another, cryptographically hashed, QByteArray.
  - c. Note that the username I/O will be handled by the QHash serialization.

### Important note:

It is acceptable for the computer to simply put an “O” randomly on the grid. You may however implement a better algorithm. If you do, you will be rewarded with up to 10 points **extra credit**.

### Actions

The actions (under the menu), in addition to the pushbuttons must drive the panels. These must be in the enabled/disabled state based on the panel they are in. In Ubuntu 14.04.1 with Unity window manager this will cause the menu items that are disabled to gray out.

<i>Menu</i>	<i>Actions</i>	<b>Main Login Panel</b>	<b>Register User Panel</b>	<b>Welcome User Panel</b>	<b>Change password Panel</b>	<b>Tic-Tac-Toe game panel</b>
<b>User</b>	<i>Register</i>	<b>Enabled</b>	Disabled	Disabled	Disabled	Disabled
	<i>Logout</i>	Disabled	Disabled	<b>Enabled</b>	<b>Enabled</b>	<b>Enabled</b>
	<i>Exit</i>	<b>Enabled</b>	<b>Enabled</b>	<b>Enabled</b>	<b>Enabled</b>	<b>Enabled</b>
<b>Game</b>	<i>New Game</i>	Disabled	Disabled	<b>Enabled</b>	Disabled	Disabled
	<i>End Game</i>	Disabled	Disabled	Disabled	Disabled	<b>Enabled</b>
<b>Edit</b>	<i>Change password</i>	Disabled	Disabled	<b>Enabled</b>	Disabled	Disabled



## Notes

1. You may use QtDesigner (from within QtCreator or, if you wish so, separately) for creating the forms and the main window. To start, simply create a new project of type **Qt5 Gui Application** inside QtCreator.
2. Using QtDesigner your widgets will be contained in \*.ui files. You can read in Qt's documentation on how to use UI files in your application: <http://qt-project.org/doc/qt-5/designer-using-a-ui-file.html>. Pay particular attention to the following sections:
  - The Single Inheritance Approach
  - Widgets and Dialogs with Auto-Connect
3. In QtCreator, while editing a QtDesigner form: Right click on a widget (i.e. a button), choose Go to slot, and select the appropriate signal you want to handle. This process will automatically create a slot in your source file.
4. Always use “Qt Designer Form Class” when creating your widgets.
5. To see examples in action, go to the Welcome tab in QtCreator, and use the “Explore Qt Examples” section. Check out the QtDesigner → Calculator Form example (and others, if you wish).
6. To draw the game grid, consider using a custom-drawn widget. Check out the documentation of QPainter: <http://qt-project.org/doc/qt-5/qpainter.html>
7. Qt's implementation of a cryptographic hash (<http://qt-project.org/doc/qt-5/qcryptographichash.html>) allows you to encrypt information passed to it in a way that is “inefficient” to recover. What this means is that given a password  $m$  you can pass it through the sha1sum algorithm and get out what's called a digest  $m'$  written  $m' = sha1(m)$ . If a hacker is able to steal your passwords file they'll be able to read the digest of every user's password. However, it should take them an extremely long time to figure out a message  $k$  which will also yield  $m' = sha1(k)$  (it's not necessary to find a  $k$ ,  $k=m$  to crack a cryptographic hash). This property means that if a user knows the password  $m$  the computer can store the digest  $m'$  and the next time the user logs in they give the computer  $m$  as the password the computer calculates  $m'$  since the *sha1* algorithm always returns the same output for a given input. If the digest the computer stored matches the digest of the password the user just gave then the computer can say with very high confidence that this person knows what the actual password is for that user and is thus the user they tried to log in as.
8. Note that other techniques are necessary to keep passwords secure in practical contexts. While you're not required to do so for this assignment you would need to give each user record a salt and key stretching. Salting would prevent two different users with the same password from having the same hash but would instead combine the salt (kept plaintext with the record) with the password to generate the hash. Salting prevents a hacker from cracking an entire password database and prevents efficient parallel cracking of a hash. Key stretching repeatedly applies a cryptographic hash function to make generation of the hash function more expensive, but still feasible, for the legitimate user but totally impractical for a hacker trying to search over the space of input passwords. You are not required to use either for this assignment. If you absolutely have to implement a password system consult the latest information on password storage.