Virginia Tech
Bradley Department of Electrical and Computer Engineering
ECE-3574: Applied Software Engineering * Fall 2012

Homework 5

Submission Details

You must submit the solutions for this homework as an electronic submissions using Scholar (under ECE3574 → Assignments → Homework 5). 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 12.04 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 5). The submitted file must be given a name in the following form: ***LAST_FIRST_hw5.tar.gz*** where LAST is your last or family name and FIRST is your first or given name. You are only allowed to make one submission. 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 5 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.

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.

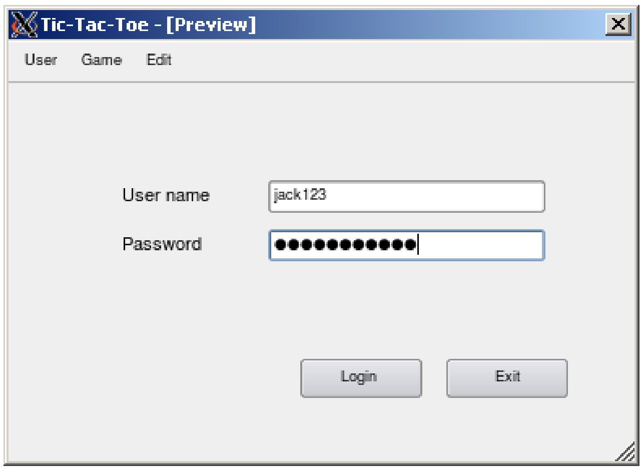
Specifications

1. The GUI will now 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 should change the password of the existing user
    e. **Tic-Tac-Toe game panel** – This panel should start a new tic-tac-toe game.

3. **NOTE**: There would be only one main window and all the panels listed above would be shown inside the main window, one at a time, based on the selection.

4. The main window will have a tool-bar with three pull down menus (“User”, “Game” and “Edit”).
    a. The “User” menu should have the following actions
        i. Register User – This action should show the “Register User Panel”
        ii. Logout User – This action should logout the current user and show the “Main Login Panel”
        iii. Exit – This action should exit the application
    b. The “Game” menu should have the following actions
        i. New game – This action should start the new game and show the “Tic-Tac-Toe” game panel
        ii. End game – This action should end the game and show the “Welcome User Panel”
    c. The “Edit” menu should have the following actions
        i. Change Password – This action should show the “Change password Panel”

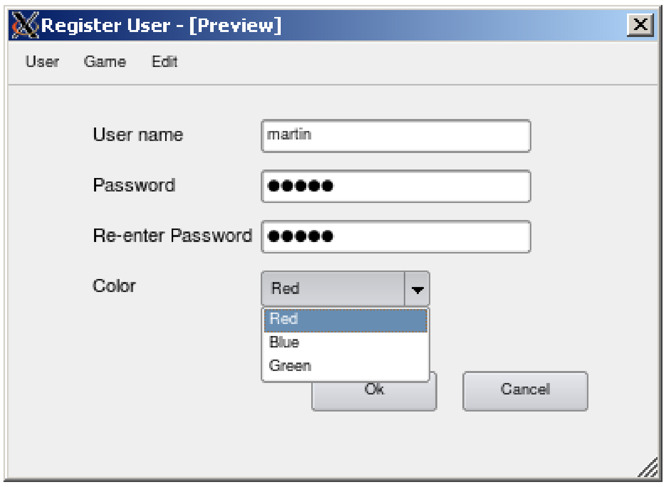
Main Login Panel



User name: jack123
Password: ●●●●●●●●●●●

1. The application should start in Main Login panel.
   a. In the Main Login panel, the user should be asked for the user name and password.
   b. Only registered users should be able to enter the application
   c. To register the users can choose the “Register User” action from the menu “User”
2. Once the user has been registered, In the “Main Login panel” when the user tries to login using the user-name and password and clicking on the “Login” button, the program should successfully log the person and show the “Welcome Panel
3. If the user clicks on “Exit” button the program should exit.

Register User Panel



User name: martin
Password: ●●●●●
Re-enter Password: ●●●●●
Color: Red

When the “Register User” action is clicked, it should show the Register user panel.
1. In this Register User panel, the user should be asked for the following details. (These should be fields)
   a. Desired user name
   b. Password
   c. Re-enter password
   d. A drop down menu with names of colors. The user should select one color.
2. The program should save the information in a **binary** file called “passwords.dat”. The users’ passwords should 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.

a. 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.

b. All password records must be serialized by QList. If you put all your records into a QList you should be able to save and read everything 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 QList<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.

c. All records must be written out in the following order and in the following format

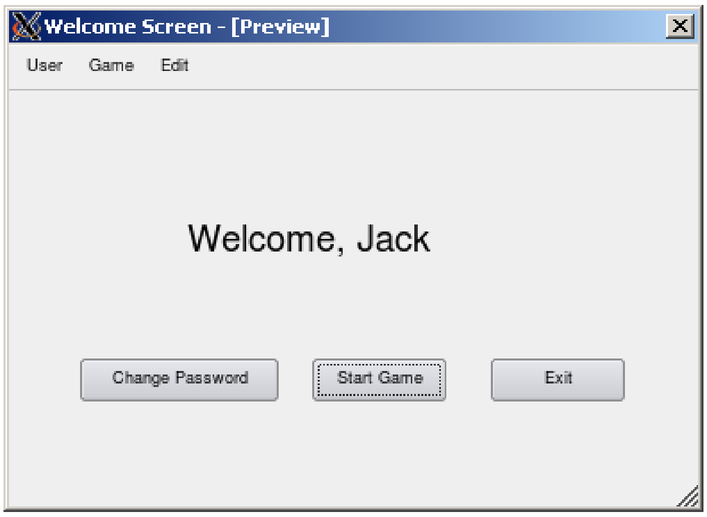
i. Name: as a QString

ii. Color: as a QString

iii. Password: convert it from a QString to a QByteArray using QString::toUtf8() and pass it through the SHA1 algorithm which results in another QByteArray.

3. When the user click "Ok", the program should check for the followings exceptions
   a. If the user already exists in the "password.dat" file, the program should 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 should throw a popup with the message "Passwords do not match, please re-enter"
4. After clicking "Ok", if all the conditions are met, the entry should be stored in the "password.dat" file and the program should go back to the "Main Login panel"
5. If the user clicks on "Cancel" the program should not store anything and go back to "Main Login Panel"

Welcome User Panel

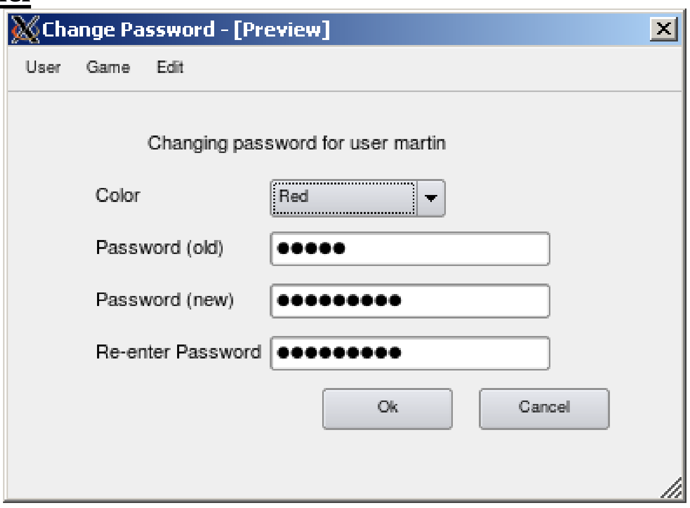


Welcome, Jack

The welcome panel should give a welcome message "Welcome, <username>" and it shall have three buttons:

1. Change Password – This button should show the "Change password panel"
2. Start Game – This button should start a new game and show the "Tic-Tac-Toe game panel"
3. Exit – This button should exit the application

Change Password Panel



Changing password for user martin

Color: Red

Password (old): ●●●●●

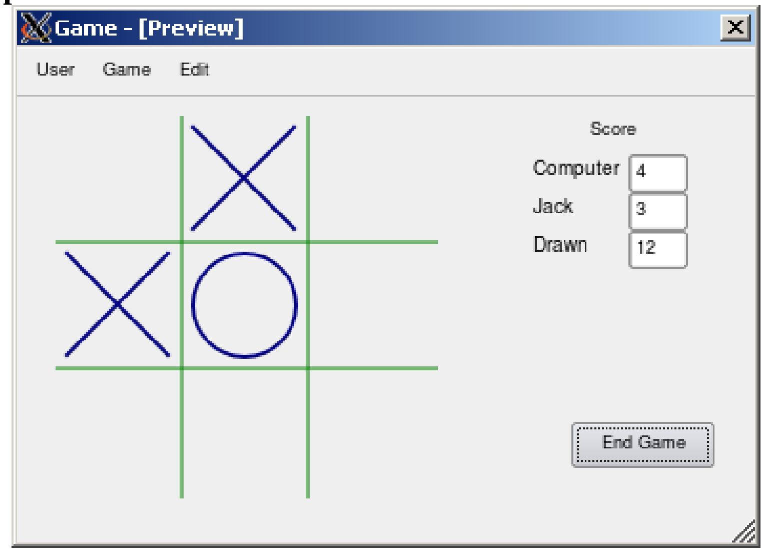
Password (new): ●●●●●●●●●

Re-enter Password: ●●●●●●●●●

Once the “change password” action (under Edit) or the “Change password” button is clicked by the user the “Change Password Panel” should be shown.

1. The following fields would be shown on this panel
   a. Old password of the user
   b. Drop down menu to select color
   c. New password
   d. Re-enter new password
2. On clicking “Ok”, the following error conditions should be checked:
   a. If the old password for given by the user does 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 should change the password in the password.dat file as well as update the player’s color and go back to the “Welcome User Panel”
4. If the user clicks on “Cancel”, the password should not be changed and the program should go back to the “Welcome User Panel”

Tic-Tac-Toe Game panel



Computer	4
Jack	3
Drawn	12

1. This panel should show a 3x3 grid for the tic-tac-toe game.
2. The grid color is up to you but the color of the “X” and “O” markers should match the color the logged in user chose during registration or while changing their password.
3. General rules for the tic-tac-toe should apply here – the person getting three in a row (diagonally, horizontally or vertically) would win the game.
4. The program should play against the user as “Computer”
5. The “user” should always make the first move and should always be given the “X”
6. The “Computer” should play second and should use “O”
7. If the game ends in draw, a popup should be shown “game drawn”. On clicking “Ok” on the pop-up message, the game should re-start (The user should make the first move)
8. If the user wins, there should be a pop-up message “Congratulations, you win”. On clicking “Ok” on the pop-up message, the game should re-start (The user should make the first move)
9. If the Computer wins, there should be a pop-up message “Sorry, you lost. Better luck next time”. On clicking “Ok” on the pop-up message, the game should re-start (The user should make the first move).
10. If the user clicks on the action “End Game” under Game menu or clicks the “End game” button on the panel, the game should end and the program should show the “Welcome User Panel”.
11. There should be a score card on the panel which should show the count of the number of games won by the “Computer”, the “User” and the number of game that were drawn
12. The user name should be shown on the scoreboard where the name “Jack” is shown in the above image.

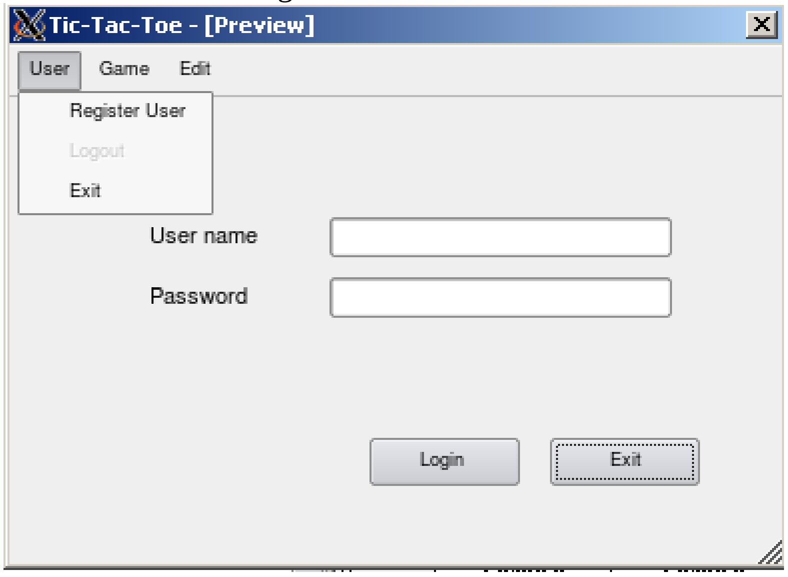
Important note:
It is acceptable for the computer to simply put a O randomly on the grid, except when the user is about to win (2 in a row). In that case, the computer should put the O in a way that tries to prevent the user's victory.

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) should drive the panels. These should be in the enabled\disabled state based on the panel they are in.

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

For example, the actions for the Main Login Panel are shown below



Notes

1. You should 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 **Qt4 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://doc.qt.nokia.com/4.6/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://doc.qt.nokia.com/4.6/qpainter.html#details
7. To submit your solution, you should use the “make dist” command in the command line (while inside your project's folder). This will archive the appropriate files.
8. Qt’s implementation of a cryptographic hash (http://qt-project.org/doc/qt-4.8/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 say *k*=*m*). 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.

Important update: You are now required to also draw a UML diagram that captures all of your classes and their relationships; also include the most important Qt classes that you use in your application. You do not need to include all the functions/methods/data members, but only those you consider important. You may use any drawing program to do that, although I recommend the **umbrello** UML editor (in Ubuntu, run "sudo apt-get install umbrello" in the console to install it, then "umbrello" to run it). You should save/export your diagram to an image or pdf document (in umbrello, from the menu: Diagrams → Export as picture) and include that in the archive you submit (Use "make dist" and then open/double-click the archive and add the new file to it). The UML diagram will be worth 6 points from your homework. I encourage you to do it before starting to write your code (sort of an higher-level design of your application) and then update it if needed while/after writing your code.

Grade breakdown

- Coding Style: 10%
- Correct usage of QtDesigner forms: 10%
- Form appearance and behavior: 54%
- Game grid: 15%
- Basic game play: 5%
- UML diagram: 6%
- Advanced game play: extra credit 10%