**PRACTICAL EXAMINATION-EXAMPLE**

This will last approximately 2.5 hours.

Below you will find a problem statement similar to what you can expect to receive during the practical examination. The problem statement will in general follow the requirements set out between Lab 5 and Lab 14, will require reading and writing from/to a file, data validation, a graphical user interface (using Qt), the implementation of layered architecture, Observer, Model/View architecture, simple drawing in Qt and may require writing specifications and tests.

Observations: 1. Solving the following problem statement completely should be possible for you in a time span of 2.5 hours.

2. You are encouraged to bring your own laptop to the exam. You are free to use your preferred IDE. Make sure your IDE is set up correctly and it works! Make sure that Qt works! If you cannot bring a laptop, you can use university computers (Qt and Qt Creator). For this, please let me know at least one day in advance that you will be using a university computer.

3. You are allowed to use Qt Designer, if you want to.

4. Only functional features are scored (source code is not scored).

5. The problem must be started from an empty workspace. You are allowed to use the following sites for documentation, but nothing else:

- <http://doc.qt.io/qt-6/>

- <http://en.cppreference.com/w/>

- <http://www.cplusplus.com/>

**Problem statement**

Write an application which simulates the development and testing of a software application, as follows:

1. The information about the development team is in a text file. Each member of the team - User has a name (string) and a type (string), which indicates whether the user is a tester or a programmer. This file is manually created and it is read when the application starts.

2. Another file contains information about the issues reported by the testers. Each Issue has a description (string), a status (can be open or closed), the reporter – the name of the person who reported it and the solver – the name of the person who solved it. These are read when the application starts and are also stored in the file by the program.

3. When the application is launched, a new window is created for each user, having as title the user’s name and type (tester or programmer). (1p)

4. Each window will show all the issues, with their description, status, reporter and solver, sorted by status and by description. (1p)

5. Only testers can report issues, by inputting the issue’s description and pressing a button “Add”. The issue’s reporter will automatically be set – this will be the name of the tester who added it. This operation fails if the description is empty or if there is another issue with the same description. The user will be informed if the operation fails. (1.25p)

6. Both programmers and users can remove issues. An issue can only be removed if its status is closed. (1p)

7. Only programmers can resolve issues, by selecting the issue and pressing a button “Resolve”. This button is activated only if the status of selected issue is open. When an issue is resolved, the name of the issue’s solver is automatically updated to the name of the programmer who solved it. (1.25p)

8. When a modification is made by any user, all the other users will see the modified list of issues. (2p)

9. When the application is finished, the issues file will be updated. (0.5p)

**Observations**

1. 1p – of

2. Specify and test the following functions (repository / service): a. Function which adds an issue. (0.5p) b. Function which updates an issue’s status and programmer. (0.5p)

3. The application must use layered architecture in order for functionalities to be graded.

4. If you do not read the data from file, you will receive 50% of functionalities 3, 4, 5 and 6.