Assignment 2

Overview

In this assignment, you will:

- Write implementation for the various GUI components that were tested in the first assignment.
- Keep track of your progress using version control.
- Run code coverage to ensure all code statements are tested.
- Use various software engineering tools to help create quality software (static and style analysis, memory leak checking, continuous integration).

Instructions

1) Go to the Git repository at http://gitlab.cs.uleth.ca/cpsc2720/GUI/asn2.
   a) This is a CS department server - you will only be able to do this on the campus network (or via VPN).
2) Set your notification settings for **the assignment repository to "Watch"** so you will receive email notification if there are any changes to repository (e.g. clarifications are added to these instructions).
   a) It is not expected that changes will be needed – this is "just in case"
3) Fork the repository so you have your own copy.
4) Set your notification settings for **your fork of the assignment repository to "Watch"** so you will receive email notifications if the build fails.
5) Set the project visibility for your forked repository to "Private".
   a) This means that other students will not access to your work.
6) Add the instructor, marker and lab instructor as members of your project with the permission "Reporter".
   a) You will be provided with their CS department user name in the lab and/or on Moodle. This is needed so the marker can grade your assignment and the lab instructor can provide assistance.
7) Setup your GitLab repository for running continuous integration for your project.
   a) Set the *Git Strategy* to "`git clone`". You will need to scroll down to find it.
   b) Set the Timeout to 5 (i.e. 5 minutes). Your CI job will be small, so this should be lots of time and will prevent any infinite loops from tying up the CI server or consuming all the disk space.

Completing the Assignment

- Create a local clone of your assignment repository.
  a. Run the command `git remote` and verify that there is a remote called `origin`.
     i. `origin` is the link to your repository of GitLab and is where you will be pushing your changes.
- Copy the following from your Assignment #1 directory to your Assignment #2 directory.
  a. your `test` and `include` directories

b. `CPPLINT.cfg`

c. `doxyfile`

- Create a `Code::Blocks` project for your assignment.
  a. You will need to add the two copied directories to your `Code::Blocks` project (i.e. "Add files recursively…")
- Implement the public/protected/private methods of the classes as specified in the header (`.h`) files using the Test-Driven Development practice.
  a. Add one unit test file (e.g. `TestImage.cpp`) to the `Code::Blocks` project.
  b. Create a `.cpp` file for the class you are testing with empty methods.
    i. Implementation files (`.cpp`) go into the `src` directory (you will have to create this directory and add it to your `Code::Blocks` project).
  c. All tests should fail at this point, as the methods have no implementation.
  d. Implement one of the public methods.
  e. Once all the tests for that method pass, implement the next method.
  f. When all the methods in the class are tested, repeat for the next class until all classes are tested.

Notes

- A `Makefile` is provided which:
  - Builds and runs a testing executable (`make tests`).
  - Checks for memory leaks (`make memcheck`)
  - Runs static analysis (`make static`)
  - Runs style checking (`make style`)
  - Runs code coverage (`make coverage`)
  - Runs all of the checks (`make all`)
- For testing `AsciiWindow`, use `diff` to compare a text file(s) created by your unit test(s) with a text file(s) containing the expected output.
  - Example expected output files are provided in `test/output/`, but you may create your own as well.
- You will not be able to run the code coverage until you have code to examine (i.e. `.cpp` files in both `src` and `test`)

Grading

You will be graded based on your demonstrated understanding of the use of unit testing, version control and good software engineering practices. Examples of items the grader will be looking for include (but are not

limited to):

- Version control history shows an iterative progression in completing the assignment. You are expected to have a minimum of eight new commits in your repository (i.e. one for each new source code file).
- Version control repository contains no files that are generated by tools (e.g. object files, binary files, documentation files)
- Memory leak checking, static analysis and style analysis show no problems with your code.
- Implementation shows understanding of software engineering principles.
- Unit tests produce 100% statement coverage (function coverage need not be 100%).

Submission

There is no need to submit anything, as GitLab tracks links to forks of the assignment repository.

- Make sure that the permissions are correctly set for your repository on GitLab so the grader has access. **You will receive an automatic 0 (zero) for the assignment if the grader cannot access your repository.**

Appendix

Updating the Assignment Files

Updating the Assignment Files

The following information is to be used in the case that the assignment is updated with clarifications or corrections.

1. Create an upstream remote to the original assignment repository from your local repository:

   ```
   git remote add upstream http://gitlab.cs.uleth.ca/cpsc2720/GUI/asn2.git
   ```

   This command creates a link from your local repository to the original assignment repository. You will not have permissions to push to the assignment repository and you will get an error if you try.

2. To get the updates from the assignment repository, you can pull them into your local repository.

   ```
   git pull upstream  master
   ```

   a. If there are any merge conflicts, you will need to resolve them.
   b. If there is a file with a complicated merge conflict (e.g. a PDF or `.cbp` file)
      i. `git fetch upstream`
      ii. `git checkout FETCH_HEAD <filename>`

3. Commit the changes to your local repository.