

Instructions: This is an open-book, open-notes exam. Laptops can be used to access online references. There are three programming problems for a total of 40 points. The exam assumes that you have the course development environment setup correctly on your laptop. You can also use the virtual machine distributed as part of milestone 3 to test your code.

To begin, go to <https://vt.instructure.com/courses/53961/assignments/343203>, and download the file `final.zip`. Unzip the file onto your machine. The zip archive contains three directories, one for each problem below. Complete each problem. Then zip up the source code (**no** build files!) for all three problems, and submit through the same link on Canvas. *This must be completed within a 30 minute window at the end of the exam.*

1. (20 points) Consider a simple Qt application that consists of a QLineEdit Widget at the top that specifies the location of a text file, below that another QLineEdit Widget that contains a string to search the file for, and below that a QPlainTextEdit Widget to display the lines that contain the string specified. These Widgets are laid out vertically.

When the user types a string to search for in the second QLineEdit and presses return, the application should read the file name specified in the first QLineEdit as a plain text file. It should then find all lines that contain the search string and display them in the QPlainTextEdit. The QPlainTextEdit should be cleared of the results from a previous search before displaying the results of a new one. If the file cannot be read then the QPlainTextEdit should also be cleared but no other action taken.

Implement the above program in the three source files `main.cpp`, `search.hpp`, and `search.cpp`. The included `CMakeLists.txt` file is setup to correctly build the application as an executable named P1. The appearance and size of the widgets can be the default, but it should be laid out as specified above, roughly like



2. (10 points) Consider the function defined in the file `filename.hpp` and implemented in the file `filename.cpp`, which can split a filename into its base and its extension.

```
/* split - split a string assuming it is a filename into its components:
   the basename and the extension.

* The basename consists of characters from the start up to but not including
  the last '.' character.
* The extension consists of characters after the last '.' character to the
  end of the string.
* Returns a pair(basename, extension).
*/
std::pair<std::string, std::string> split(std::string filename);
```

Write a Catch based unit test in the file `filename_test.cpp`. If any test fails it should be because of a bug in the function. The included `CMakeLists.txt` file is setup to correctly build the test as an executable named P2.

3. (10 points) Consider a program that consists of two threads. The main thread should print "Go " to standard output, followed by the second thread printing "Hokies!" followed by a newline. This sequence should run 10 times at which point the application should exit successfully. The included `CMakeLists.txt` file is setup to correctly build the application as an executable named P3 from the file `main.cpp`. You can only use the C++11 standard library.