# COMP220 - Lab 4 & Homework 4

Fall 2015

**Abstract**

For this lab/hwk pair you'll extend the equality library discussed in class for the StanfordCPP library Stack and Queue classes. The interfaces for these classes is listed on pages 212 and 218, respectively. The initial equality library, with Vector implementations, can be found in the course home directory; the file *scpplibequals.zip* contains the the library (header and implementation) and Google Tests.

## 1 The Assignment

For *lab4* get started on implementing *operator==* for the StanfordCPP Stack and Queue. You should work our process: document, declare, stub, write gtests, then, and only then, implement. Implementation will be a bit trickier than it was for vectors because stacks and queues are not random access. You have to deconstruct the container in order to get at all the contents. You'll have to decide what this means for the signature (do you pass by value, reference, or constant reference?). At the end of the lab period, submit your work as assignment *lab4*. If you did not finish the two operators, then complete them by **Monday 9/21** and submit your source code as assignment *hwk4*. The lab grade is about the *quality and quantity of your progress*. The homework grade is about the quality and correctness of the final product.

## 2 Notes on Template Libraries

Writing and working with template libraries is a bit different. For non-template libraries we would include the header in source code and compile the implementation for linking with other compiled objects. Templates cannot be compiled until you've used them in code, until a programmer has specified the exact values of the template parameters. This causes a key problem for our library design methodology. We can't compile the template implementation file so how do we provide complete definitions to the compile without linking? The answer is a bit unsatisfactory. Write your header and implementation files like you usually do, but *include the implementation file at the end of the header file*. This breaks the "don't #include cpp files" rule. This is the one exception. By doing this we can #include the header file wherever we need the template and also bring along the complete definitions. When we compile the code that uses our template library header, the compiler will use the template as needed and compile the code that results. You can see all of this at work in the *scpplibequals* library starter you've been given for this lab/hwk: the cpp is included at the end of the header, the header is included in the Google Tests, the Google tests utilize the template code which is compiled as needed when the Google tests are compiled.

### 2.1 Using your Template Library in other Projects

To use the template library you must direct your compiler towards the header file. One way to do this is to copy the header into the program's working directory. This way is bad practice. We'll use this equality library a lot and don't want to keep copying it. The better fix is what you did for the StanfordCPP library. Add the directory containing the header and cpp file to the search path for the compiler.