- *core*, which is where you'll normally write most of the C++ source code for your projects. The idea is that the *core* directory contains the "core" code for your project (e.g., data structures, algorithms, user interface functionality, etc.), without a *main()* function, unit tests, or any other outer "shell."
- *app*, which will generally contain only a single source file that defines the *main()* function for your project. Ordinarily, you'll want *main()* to do nothing but call into code from your *core* directory or other code that we've provided; quite often, your *main()* function will be just a couple of lines of code.
- *exp*, which will contain a "playground" program in which you can experiment with code from your *core* directory without affecting the main program you're writing. In this directory, you'll write a separate *main()* function, along with whatever other experimental code you'd like to write. This can be handy for testing or experimenting.
- *gtest*, which will contain unit tests written for the Google Test framework. Already present will be a file *gtestmain.cpp* that you should not modify. Simply create new source files with unit tests in them and they will automatically be picked up on the next build

The requirements

Notice that three files are already present in the *core* directory: *String.hpp*, *String.cpp*, and *OutOfBoundsException.hpp*. Your goal in this project is to implement the *String* class declared (and described in comments) in *String.hpp*. There are a few requirements to be aware of:

- You are not permitted to use any standard library functionality – this includes both the C Standard Library and the C++ Standard Library – since the goal is to exercise your skills at writing lower-level C++ code. (This is the kind of problem you'll need to be adept at solving, if you want to succeed in this course, so better to test and solidify those skills out now.) Note that even including a standard header file, such as *<iostream>* or *<cstdlib>* violates this rule.
- Implementations of all member functions must be written in *String.cpp*, though you will probably need to declare some additional private members (member variables and/or member functions) in *String.hpp*.
- Your implementation must not leak or mismanage memory, even in situations when exceptions are thrown from your member functions. (This includes both exceptions that you throw yourself, as well as exceptions arising from the things you do – note, for example, that the *new* operator throws *std::bad_alloc* when there is not enough memory to allocate what's been asked for.)

  You'll notice, too, that there is a file called *StringTests.cpp* in the *gtest* directory, which provides an example of how each of the member functions should behave. All of these tests should pass (and exhibit no memory-related issues) when your implementation is complete. You can feel free to add any additional unit tests you'd like, but resist the temptation to modify the existing ones.

How do I debug my program without using iostream?

You may have become accustomed to doing a lot of your debugging by printing things to the standard output by including, for example, the <iostream> header from the C++ Standard Library. Since you aren't permitted to include standard headers in this project, you might be wondering how to debug your work. In truth, the kind of work you're doing in this project probably doesn't lend itself to printing-based debugging, anyway, since the kinds of things that are likely to go wrong will have symptoms that will be difficult to diagnose by simply printing things out. For that reason, debugging your work in this project is best done with the LLDB debugger.