Due Date: Friday, January 27, 2017 - 5:00 pm
Your name:

INTRODUCTION

This lab touches on a number of topics. First, you will be asked to extend a virtual class that does matrix operations. Second, you will be asked to write a series of unit test functions for your extended class. You will be required to build your library and unit test functions using CMake. For those taking the class for more than two credits you will be asked to build a docker container that builds and runs your test executable. You will find the Canvas website for this class useful to answer your *How do I?* questions.

GETTING STARTED

Begin by logging into cees-rcf `ssh -Y cees-rcf`. Begin by downloading the lab. The easiest way to do this to type `curl`. Next change into the git directory you created in your last lab and untar the tarball you downloaded. Assuming you downloaded the tarball into your home directory you would type `tar xf  /lab2.tgz`.

PROBLEM 1: EXTENDING A CLASS [5 PTS]

In class we have now shown examples of inheritance in both python and C++. If you look into your lab2 directory you should a directory called `Src`. In that directory you will find two base classes defined in (`vector.h/vector.cc` and `matrix.h, matrix.cc`). The first base class `vector` defines an abstract class that defines some linear operations that can be applied to vectors such as addition and scaling. In addition you will find a concrete implementation of the vector class, `vector1d`. The main purpose of this class it remind you on how inherit a class in C++ and override its virtual members. Your first job is to inherit two concrete classes (a full and sparse matrix) from the abstract `matrix` class. I've already set up a rule to compile a library (`matVec`.

Take advantage of the `ptr` and `cptr` function in the vector classes. Once you've written your classes you can compile them by typing `make lib`

PROBLEM 2: UNIT TESTING [5 PTS]

Your next job is to write some unit tests for your matrix class. Open up the file `unitTest.cc`. If you type `make test`, the `unitTest.cc` file will be compiled and linked with your library. I've written a series of unit tests for the vector class that test to see if my initializer, addition, scale, and dot functions work correctly. Your job is to write a series of unit tests for your sparse and full matrix test. Make sure to test

- initializers
- applyForward
- applyAdjoint
- Adjoint is actually your adjoint of your forward

PROBLEM 3: CMAKE (5 PTS)

Distributing code to other people is one of the most challenging portions of software development. The makefile provided is specific to the way CEES is setup. Your job is to replace my build mechanism with something that attempts to figure out how a given computer is setup to create a customized makefile. There are several different approaches to build customized makefile. Your job is to create a CmakeLists.txt file to build your library and run your unit test code. In addition create a file called `RUNTHIS` that invokes cmake and compiler correctly on CEES. A couple hints: the default CEES environment uses a very old gcc compiler that does not support C++11, you will need to check to make sure the chosen compiler supports C++11 and specify a compiler that will work on CEES using an environment variable. Once you've

completed this portion of the lab, add its contents to your git repository `git add .`, commit it `git commit -m "Finished lab2-part3"`, and push `git push` to the server.

If you are taking this class for 2 credits your can stop now.

PROBLEM 4: DOCKER (10 PTS)

The previous section illustrates one of the challenges with distributing code. If you are using anything even remotely new, or non-standard it can become challenging for a user/system admin to setup the environment needed to compile and run your code. Virtual machines, and later containers, became an answer to how to guarantee that that developer's environment is the same as the consumer's environment. For this portion of the lab we will be using docker. With docker you will describe on how to build an operating system from scratch that can compile your library.

If you are doing this assignment in the computer lab you will find docker already installed. Otherwise install it on your laptop/local machine. Next, clone your git repository to your local machine. Your job is to make a Dockerfile that builds your code. `Hints`

- Download a linux operating system
- Install the g++ compiler
- Install boost development environment
- Install cmake
- Add your source code
- Compile your source code

Once you are done and your Cmakefiles.txt file to your repository, check it in, and push it so your TA can grade it.

c