Programming Exercise 7

EE312 Fall 2013

To complete during recitation, the week of 10/28

TWO POINTS

You will be provided with a simple program that contains at least one error. Your challenge during this exercise is to design a test program that confirms the existence of that error. You are encouraged to pursue a test program that, if all tests were passed, would confirm the absence of errors.

STEP1: The Vector Case Study

The Vector class developed in lecture includes a Constructor, an operator[], a size function and push_back and pop_back functions. Design at least five tests for this class. While it is not necessarily a good idea to use every function in every test, it is essential to call every function at least once somewhere in your test suite. To successfully test the Vector, you will need to

- Create Vector objects of different sizes
- Populate the Vector objects with different values
- Use the size, push_back and pop_back functions on your Vector objects to ensure correct behavior
- Make copies and perform assignments to Vectors.

As you write your test functions you can use only the public interface to the Vector (you cannot directly access the private components of the struct). I'm aware of one error in the design and implementation of the Vector, and you should be able to reveal at least this error through your testing. Hopefully the error I know about is the only error in the program. It is not necessary to fix the error, only to confirm its existence. To receive credit for this exercise you must commit your test functions to the repository. The test functions should be named test1(), test2(), test3() etc. and should be placed into the file main.cpp.

STEP2: Testing the BST

This part of the exercise is left for another day. However, if you have additional time to think about how you're modify your test suite for the BST data structure and/or if you have time to discuss how to test the BST with your recitation group, then that's a great thing to do. Please do recognize that it is possible (and very often a good idea) to write the test cases **before** you write the actual software you'll be testing. So, even though we've not yet built the BST, it's definitely not too early to think about how we'd test it.