

CS 14: Data Structures and Algorithms

Lab 2

Peter H. Fröhlich
phf@cs.ucr.edu

Wagner Truppel
wagner@cs.ucr.edu

Out on: January 13, 2003

Due by: January 17, 2003

Remember to follow the instructions and policies for assignments given on the course website. The assignment consists of a **programming** exercise to be completed **in lab** as well as a **programming** exercise and to be completed **at home**.

The “in lab” parts are due at the **end** of your **lab section**. The “at home” parts are due **before 8:00 pm** on the date given above. You have to use the department’s electronic turnin service for **everything**.

1 In-Lab: Debugging

You can download an archive for this lab from the course website. Inside, you will find a file `buggy.cpp` which contains a number of bugs. The program **should** produce the following output when invoked (edited to fit, should be one line each):

```
[0][1][2][3][4][5][6][7][8][9]
 [10][11][12][13][14][15]
 [15][14][13][12][11][10][9][8]
 [7][6][5][4][3][2][1][0]
```

When you compile and run the program, you will notice quite a different output. Try to fix the program by making a minimal number of changes to it, do **not** rewrite the whole program from scratch. Note that even when the output is

exactly as described above, there might still be another bug...

The reading you were assigned to do for this lab, the chapter on **debugging** in *The Practice of Programming*, should prove helpful here. If you don’t know how to use the debugger, ask your teaching assistant.

2 At-Home: Counters

The archive you downloaded also contains a file `counter`. It defines an *abstract base class* describing the interface of the abstract data type (ADT) `Counter`, which is (in turn) partially described in comments. There’s also a program `testcounter.cpp` which tries to create an object of the class `SimpleCounter` implementing the `Counter` interface. Your task is to develop a class `SimpleCounter` that passes the tests performed in `testcounter.cpp`.

Hints: Your class should be derived publicly from `Counter`, and should use exceptions to signal errors. Errors occur whenever the value of a `Counter` would be outside the bounds it was created with. Throwing the exception `std::domain_error` is as good a choice as any for now.