CSC 211 Lab 1: Fixing a program

Spring 2017

About this lab

In this lab, you will explore several tools that you will use throughout the semester. You will download a program written by your instructor, identify a bug in it, and fix that bug.

Find a partner

You will do this lab with a partner, and ideally, you will work with that partner for the first programming assignment. So, *find a partner*.

Logging into the lab machines

Step 1:

You should have received an email with your lab username and password. Take turns (with your partner) logging into your accounts. At the very least, make sure one of you is able to log in.

You may use your own computer if Docker is working properly from the previous lab.

Step 2:

Open Docker if it is not running (the whale icon in the menu bar will tell you its status).

Open Terminal if it is not running (if you don't see it, hold the `command` key and hit space, then type Terminal and hit return).

If you are on a lab machine, or didn't get this working in the previous lab:

Create a folder on the Desktop called CSC211.

Type `echo "alias d211=\"docker run -t -i -v ~/Desktop/CSC211:/home/student/data ndaniels/csc211-dev\"" >> ~/.bash_profile`. This writes the `alias` command into the file `.bash_profile` which ensures that it will make the `d211` alias available whenever you open up a Terminal.

Close and re-open the terminal window.

Downloading and compiling the lab code

Type `d211` which should launch the CSC211 docker environment.

Type `cd data` which will put you in the folder that's shared with the Desktop.

Type `git clone https://github.com/csc211/lab1.git` which will download this lab.

Git is a tool for *revision control*. It lets you keep a record of changes made to a program. Git, or other programs like it, are used by virtually all professional programmers.

Use the `cd` command to change into the directory that was just downloaded (note that we will no longer tell you the exact syntax for commands you have already used).

Type `git log`. You should see a history of changes to this repository. You also now know your professor's personal email address. Please don't use it.

Launch the Atom text editor (but do not close the Terminal window). Navigate to the CSC211 folder, and open the csc211-lab1 folder as a project in Atom. Click on `stats.cpp` and you should see the code.

Switch back to the Terminal (but don't close Atom). You will move between these two tools often.

Type `clang++ -o stats -Wall -Wextra stats.cpp`

This invokes the `clang++` compiler with several arguments:

- `-o stats` says that the resulting *executable program* should be called `stats` rather than the default `a.out`.
- `Wall` says to warn you of "all" problems the compiler can find with your code.
- But because "all" isn't really all, `-Wextra` says to add extra possible warnings.
- `stats.cpp` is the source code file that you're compiling.

Running the code

Type `./stats 1 22 0 5` and note that, for these inputs, the `stats` program tells you the largest number you entered.

Experiment with a few other numbers.

Being paranoid

Sometimes, programs that seem correct can hide dangerous bugs.

Type `clang++ -o stats -Wall -Wextra -g stats.cpp`

The `-g` tells the compiler to include *debugging information* which will be useful. In general, for this class, you will want to use the `-g` flag.

Now, run `valgrind ./stats 1 22 0 5`. You may want to make the Terminal window larger.

Note the error summary: valgrind has detected a memory error. Now look a bit further up, and you should see the specific error (it should start with "conditional jump or move...").

Try to make sense of this error, and then draw out, by hand if you like, what the arrays in the program look like. Can you figure out why this error is occurring?

Fixing the code

Try to fix `stats.cpp` so that it still compiles, runs correctly, and valgrind does not report any errors.

Beyond just making the program *work properly*, can you think of a way to simplify it? *Hint:* by introducing one more variable in `main()`, can you make the logic easier to follow?

Committing your changes

Type `git add stats.cpp`.

Type `git config --global user.email "<your email address>"` omitting the angle brackets.

Type `git config --global user.name "<your name>"` omitting the angle brackets.

Type `git commit -m "Fixed off-by-one error"`.