CS 521/621
Homework 3
Automated Bug Repair

Due: **Thursday, Nov 29, 2012, 9:00 AM EST** via Moodle. You may work with others on this assignment but each student must submit his or her own write up, clearly specifying the collaborators. The write ups should be individual, not created jointly, and written in the student's own words. Late assignments will not be accepted without **prior** permission.

Overview

The goal of this assignment is to learn about Genprog, an automated bug repair technique that takes a program with a set of passing and a set of failing test cases, and automatically produces a small change to the program that makes it pass all the tests.

The assignment consists of:

1. Reading a small part of a research paper that describes the technique.
2. Setting up a virtual machine to run Genprog on a buggy program that contains a simplified version of the bug that took down all the Zune music players on Dec 31, 2008. Running Genprog and analyzing its output.
3. Answering problem questions about Genprog and the Zune bug.

Resources

- The paper describing Genprog: "Automatically Finding Patches Using Genetic Programming" by Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest: `http://www.clairelegoues.com/docs/legoues-icse09.pdf`
- Virtual Box. Because the software you will be using is a research prototype, the class staff has created a virtual machine with that software installed, rather than making you deal with installing it yourself.
- The virtual machine with Genprog installed. Download the virtual machine `genprog-vm.ova` from `http://goo.gl/6TkZg`
  This files is over 600MB, so make sure you have a good network connection and allow for adequate time.
- Genprog is already installed on the virtual machine. See below on how to run it.

Setup

1. Read the first two sections (Introduction and Motivating Example) of "Automatically Finding Patches Using Genetic Programming" by Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest.

2. Download Virtual Box (you'll need it to run the provided virtual machine. Go to https://www.virtualbox.org/wiki/Downloads and download the Virtual Box for your OS. It's free and you may find this a useful tool in the future. If you have any problems installing or running Virtual Box, that website has an extensive manual and other help.

3. Make sure you downloaded the virtual machine itself: `genprog-vm.ova`.

4. Start Virtual Box, go to the `File` menu and select `Import Appliance`. Then select `Open appliance` and select the `genprog-vm.ova` file and click `Open`. Next, click `Continue` and then `Import`. Finally, click the green arrow labeled `Start` to start the virtual machine and wait for it to boot up. You can select the defaults for whatever questions the boot process asks.

5. Log into the machine:
   username: `genprog`
   password: `password`

6. The are several files of interest:

   `zune/zunebug.c` is a buggy program. (The bug in this program is a simplified version of the bug that took down all the Zune music players on Dec 31, 2008.)

   `zune/test.sh` contains 20 test cases that pass (p1–p20) and 4 test cases that fail (n1–n4). `zune/output.*` files contain the expected outputs for these tests.

   `configuration` is a configuration file for Genprog, but you won't need to modify it.

Problem

`zune/zunebug.c` contains a bug. Of its tests (in `zune/test.sh`), 20 pass, but 4 do not, because of the bug. Genprog is going to try to automatically repair this program by producing a similar program that passes all the tests. Take a look at `zune/zunebug.c` and see if you can spot the bug.

Run:

```
cd zune
$REPAIR configuration --minimization
```

Genprog will run and produce `sanity.c`, `repair.c`, and `minimized.c`, which are all modified versions of `zunebug.c`:

- `sanity.c` is a properly formatted, still buggy version.
- `repair.c` is a repaired version.
- `minimized.c` is a repaired version with fewer changes (Genprog uses delta debugging to minimize the repair).

Running this command will take a while, around 20 minutes. Genprog is magic. It's magically repairing a buggy program for you. Magic takes time.

Examine `repair.c` and note the differences with zunebug.c (you may want to run

```
diff --side-by-side sanity.c minimized.c > compare
emacs compare
```

When you want to get out of emacs, press `CTRL-x`, `CTRL-c`.

Deliverables

You should submit either `writeup.pdf` or `writeup.txt` with answers to the following questions:

1. Genprog takes a program that fails one or more tests and produces a similar program that passes those tests. Rather than exploring all programs up to a certain size difference from the original buggy program, Genprog relies on genetic algorithms and may explore some programs that are quite different from the original program, while not considering some others that are very similar. To explain why Genprog makes this choice, describe why Genprog doesn't explore (a) only all very similar programs (e.g., ones only up to 5 characters different from the original), or (b) all slightly similar programs (e.g., ones up to 10 statements different from the original). Note that the answers to (a) and (b) are different.
2. Name two concrete ways in which Genprog makes its search space tractable (makes it smaller).
3. How does Genprog make sure that the repaired program it produces is similar enough in behavior to the original buggy program. In other words, how do we know Genprog won't produce a completely different, unrelated program that just happens to pass the previously-failing test?
4. Why does it make sense for Genprog to apply delta debugging to the repair it produces? You may want to look at the difference between `repair.c` and `minimized.c` to answer this question.
5. Describe the bug in `zunebug.c` and how Genprog suggests fixing it.

Acknowledgments

Claire Le Goues was instrumental in designing this homework assignment.