CS 100
Lab 3: Vi and GDB

Reminder: find a *new* partner for this lab (50% grade reduction for working with a person ever worked before).
Today you are allowed to use 1 computer per person (2 for a team).
Lab objective: practice working with Git, Vi and GDB.

Grading: 45% each requirement, 10% attendance.

1. **Vi**: We want you to become familiar with the text editor Vi. If you are unfamiliar with it, take the first 15-30 minutes to learn the basics. There is a built-in tutorial. To open it, open up a terminal, type 'vimtutor', and hit enter. The 'j' key will allow you to move the cursor down to reach the first lesson. Try to get through at least lessons one and two at this time. In order to better understand the tutorial, do not simply read it, but practice the commands as you go through each section.
Starting today you are required to use only Vi (vim) to write programs in the labs.

2. **GDB**: Debugging with cout/printf() statements works okay when the programs are very small. When code becomes larger and more complicated, debugging in this fashion becomes untenable. Instead, special programs, known as debuggers, exist. They come equipped with tools to make the process easier and more efficient. Linux comes with a command-line debugger called gdb. To begin learning how to use the program, read the tutorial at:
http://undergraduate.csse.uwa.edu.au/units/CITS2230/resources/gdb-intro.html.

For this lab, create 1-3 programs (using Vi!), all of them buggy to some extent, which your partner will debug using gdb. **Each partner** must write **at least 150 lines of code**: either one program of 150 lines, or two programs of 75 lines each, or three programs of 50 lines each. Enter several **run-time** bugs (8-9 for one program, 4-5 each for two programs, and 2-3 each for three programs) into each of these programs which should include segmentation faults, infinite loops, bad i/o, incorrect initializations, incorrect logic, incorrect results of computations, etc.
DO NOT ADD COMPILATION ERRORS (i.e. missing semicolons or any other kind of bad syntax)!
Switch the programs you created with your partner and use the features introduced in the tutorial above to track down the errors in your partner's program(s).
To finish the lab and get credit, you will need to show that you have finished the debugging tasks above. For each program, **write comments on each line you fixed a bug**. Leave one bug unfixed and demonstrate the TA how you can find it using GDB (put a breakpoint, use trace, step, backtrace, etc.).
For those interested in a more in-depth tutorial about gdb, look into this link: http://dirac.org/linux/gdb/.