CS33, Spring 2016
Lab 0: Mini Data Lab
Assigned: March. 28, Due: Monday, Apr. 4, 11:59PM

1 Introduction

Lab 0 is the simpler version of Lab 1: Data Lab. For this lab, you need to implement a single "programming puzzle" described in Part B. The purpose of this warm-up assignment is to make sure all students:

1. can access their SEAS account remotely

2. are comfortable using basic Linux commands and programming editors

3. are familiar with Data lab structure and debugging techniques

2 Part A: Setup Your Environment

We will use the shared directory `/w/class.1/cs/cs33/cbin/` on SEASnet machine `lnxsrv.seas.ucla.edu` to distribute the files needed for each lab. Log in to your account on SEASnet machine and go to the shared directory using this command:

```
unix> cd /w/class.1/cs/cs33/cbin
unix> ls
```

The command `ls` will show a list of files and folders in current directory. You should be able to see two folders `lab0-handout` and `datalab-handout`. Both folders have the same set of files, all required for running and testing the Data lab. In both assignments, the only file you will be modifying and turning in is `bits.c`. The `README` file contains additional documentation for each file in these folders. Open and compare the `bits.c` files in two folders. While you are expected to implement 10 puzzles in Lab 1, you will implement a single puzzle in Lab 0 just to get familiar with the SEASnet machine and Datalab infrastructure. Start by copying the lab0-handout to your local SEASnet directory:

```
unix> cd ~/
unix> cp -r /w/class.1/cs/cs33/cbin/lab0-handout .
```

You could either code and debug on the SEASnet machine or copy the handout to your personal machine and code and debug locally. For either case, refer to `setup_environment.pdf` for more information.

3 Part B: Get Familiar with Data Lab structure

As mentioned in Part A, you will only modify and hand in the bits.c. Looking at the file bits.c, you will notice a C structure studentID into which you should insert the requested identifying information about yourself. Do this immediately so that you do not forget.

Your assignment is to implement the function ezThreeFourths(x) = (x*3/4) (described in the class) using only *straightline* code (i.e., no loops or conditionals) and a limited number of C arithmetic and logical operators:

```
 !  ~  &  ^  |  +  <<  >>
```

We have included some autograding tools in the handout directory — btest, dlc, and driver.pl — to help you check the correctness of your work.

- **btest:** This program checks the functional correctness of the functions in bits.c. To build and use it, type the following two commands:

  ```
  unix> make
  unix> ./btest
  ```

  Notice that you must rebuild btest each time you modify your bits.c file. Check the file README for documentation on running the btest program.

- **dlc:** This is a modified version of an ANSI C compiler from the MIT CILK group that you can use to check for compliance with the coding rules for each puzzle. The typical usage is:

  ```
  unix> ./dlc bits.c
  ```

  The program runs silently unless it detects a problem, such as an illegal operator, too many operators, or non-straightline code in the integer puzzles. Running with the -e switch:

  ```
  unix> ./dlc -e bits.c
  ```

  causes dlc to print counts of the number of operators used by each function. Type ./dlc -help for a list of command line options.

- **driver.pl:** This is a driver program that uses btest and dlc to compute the correctness and performance points for your solution. It takes no arguments:

  ```
  unix> ./driver.pl
  ```

  Your instructors will use driver.pl to evaluate your solution.

4 Handin Instructions

- Make sure it compiles, passes the `dlc` test, and passes the `btest` tests on the class machines, i.e. lnxsrv.seas.ucla.edu.

- Make sure you have included your identifying information in your file `bits.c`.

- Don't include the `<stdio.h>` header file in your `bits.c` file, as it confuses `dlc` and results in some non-intuitive error messages. You will still be able to use `printf` in your `bits.c` file for debugging without including the `<stdio.h>` header, although `gcc` will print a warning that you can ignore.

- Remove any extraneous print statements.

- Submit your `bits.c` file to CCLE where indicated under Lab 0.