15-122: Principles of Imperative Computation

Lab Week 2 — Tom Cortina, Nivedita Chopra

Setup: Copy the lab code from our public directory to your private directory:

```
% cd private/15122
% cp -R /afs/andrew/course/15/122/misc/lab-integers .
% cd lab-integers
```

You should write your code in a new file, `reverse.c0`, in the directory `lab-integers`.

Grading: Finish task (1.a) for partial credit, and finish both (1.a) and (1.b) for full credit. We encourage you to do the bug diaries task, during or after the lab.

Manipulating integers with a loop

For these two tasks, you'll need to use a loop to manipluate integers. We can identify two ways of manipulating integers in C0:

- The mathematical operations of multiplication (`a * b`), division (`a / b`), modulo (`a % b`) addition (`a + b`), subtraction (`a - b`), and negation (`-a`).
- The bitwise operations bitwise-and (`a & b`), bitwise-or (`a | b`), bitwise-xor (`a ^ b`), bitwise negation (`~a`), left shift (`a << b`) and right-shift (`a >> b`).

We don't always think about these operations as distinct categories! Sometimes, for instance, we think about `a << b` as the mathematical operation $a \times 2^b$. But for this assignment we will make the distinction.

(1.a) In `reverse.c0`, write a function `reverse_dec` that reverses the decimal digits in a nonnegative number with at most 7 decimal digits. Treat a number with fewer digits as if it has leading zeroes.

Give appropriate contracts (if any); use only *mathematical* operations on integers: `* / % + -`

```
% cc0 -d reverse.c0
--> reverse_dec(1512200);
22151 (int)
--> reverse_dec(42);
2400000 (int)
--> reverse_dec(3749);
9473000 (int)
```

You can test your code against our test cases by running `cc0 -d -x reverse.c0 test-dec.c0`

(1.b) In `reverse.c0`, write a function `reverse_hex` that reverses all the hex digits of any integer:

Give appropriate contracts (if any); use only *bitwise* operations on integers: `& | ^ ~ << >>`

```
% cc0 -d -lutil reverse.c0
--> int2hex(reverse_hex(0x195D3B7F));
"F7B3D591" (string)
--> int2hex(reverse_hex(0xC0CAFE));
"EFAC0C00" (string)
--> int2hex(reverse_hex(16));
"01000000" (string)
--> reverse_hex(int_min());
8 (int)
```

You can test your code against our test cases by running `cc0 -d -x reverse.c0 test-hex.c0`

Bug diaries

A Bug Diary is an electronic notebook where you can maintain a record of errors that you make while programming – perhaps, after this lab, you have a few fresh bugs you've encountered and then fixed! The idea is that every time you successfully find and fix a bug in your code, you make a note of it in your Bug Diary. The next time you're stuck bug-hunting, you can look through the bugs in your Bug Diary. If you find your bug there, it saves you some time.

You will not be graded on the Bug Diary, but we're encouraging all our students to create and maintain them. We did some research last semester on the bugs that students make in this class, and noticed that a lot of them are related. We think a Bug Diary will help keep you from getting stuck on the same bug twice.

Since we're doing research on Bug Diaries, we'd really appreciate it if you would create your bug diary with Evernote (see below) and share your Bug Diary with the researchers. This is so we can learn more about how bug diaries are useful – we'll only use the information in your shared bug diaries anonymously to make the course better in this and future semesters.

Setup for the bug diary

(a) If you already have an Evernote account, log into your account, and skip ahead to Step 5.

(b) Go to `https://evernote.com/` and click the "Sign Up Now" button. Enter your email address and a password to create your account.

(c) You'll reach a welcome screen: Hit "Continue" there.

(d) Now it will ask you to "Create your first note." Click that button, and you'll reach a screen with a side bar of buttons, and a blank space to type up a note (optional : go through the tutorial and learn how to create a note).

(e) Click on the button in the sidebar that says "Notebooks." Create a new notebook by clicking on the button at the top right corner (which looks like a notebook with a '+' sign) of the frame titled "Notebooks" (to the right of the sidebar)

(f) Title your notebook as "Bug Diary" and click "Create notebook."

(g) Click the "Share" button (located directly below the heading that says "Bug Diary," and "Created by . . ."). Enter the email address "`nivea.chops@gmail.com`. We only need to be able to view the notebook, so you can also change "Can edit and invite" to "Can view" (at the bottom of the frame).

(h) Hit "end" and you're done. Thank you for helping us make 122 better!

Using the bug diary

(a) When you fix a bug, create a new note describing the bug (remember to change the notebook to "Bug Diary" at the top left corner). Here's a template you can use:

    `http://tinyurl.com/bugnote`

(b) When you're stuck on a bug, go through all the notes in your Bug Diary. See if any of those bugs could be your current bug. If it is, go fix your code :)

    It will help our research if you also adding something in the note that tells us it helped you (e.g. "Helped me solve bug X" or "Seen again on date D")!