cs154: Introduction to Computer Systems
Spring 2020

Homework 1
(Assigned Apr 6th)
Due Apr 13th 11:59pm

Submit your answers by adding and committing one file with your answers into the `hw1` directory of your `CNETID-cs154-spr-20` svn repository[1].

The file should be named either `hw1.txt` or `hw1.pdf` for answers written in a plain ASCII text file, or PDF file, respectively. PDFs of scanned hand-written pages must be clearly legible, and must not exceed 5 megabytes. No other file formats, or filenames are acceptable, and no files besides `hw1.txt` or `hw1.pdf` will be graded. There is no need or benefit to including your name or CNetID in your HW filename. Not following directions may result in losing points.

(Q1) 10 points

Consider the following self-contained program:

```
#include <stdio.h>

int main() {
  int foo;
  char endian;
  foo = 0;
  endian = 0;
  printf("hello world, from a %s-endian machine\n",
         endian ? "big" : "little");
  return endian;
}
```

Figure out what should be used in the assignments to `foo` and `endian` (instead of the current initializations to zero) for the program to correctly describe the endian-ness of the machine it is running on. The assignment to `foo` should be a constant, and the assignment to `endian` should be an expression involving `foo`. No other variables or lines of code are needed. Your program should not assume anything about word size, but you can assume int contains more than one bytes, and char has only one byte.

Write your answer like:

```
foo =
endian =
```

To get full credit you need to explain your reasoning in one or two sentences: why does this code work?

(Q2) 8 points + 2 bonus points

For questions marked with bonus, you won't lose any point if you skip them, or answer them incorrectly, but correct answers can help make up for the lost points in the SAME question (Q2 here). However, the total points will not exceed the regular upper bound. For example, if you get 7 points from the regular parts (A, B, C, D here), and 2 points from the bonus one, then your total points will be 8.

For each of the following, create C expressions involving a given `int x` that **evaluate to 1 if and only if** the condition is true, and 0 otherwise. A "nibble" is half a byte (4 bits). Follow the bit-level integer coding rules in the textbook, **with the additional restriction** that you may not use equality (==) or inequality (!=) tests.

(Starting next page...)

[1] see `https://classes.cs.uchicago.edu/current/15400-1/svn.html`

A. At least one bit of `x` is 1.

B. At least one bit of `x` is 0.

C. Any of the bits in the least significant nibble of `x` is 1.

D. Any **odd** bit of `x` is 1. The least-significant bit is an even bit. For **this part** you can assume an `int` is 32-bits.

(Bonus) E. Any of the bits in the most significant nibble of `x` is 0.