**cs154: Introduction to Computer Systems**
**Autumn 2019**

**Homework 1**
(Assigned Oct 2nd)
**Due Oct 7th 11:59pm**

Submit your answers by adding and committing one file with your answers into the `hw1` directory of your `CNETID-cs154-aut-19` 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.

## (1)

Consider the following self-contained program:

```
#include <stdio.h>

int main() {
  int endian, foo;
  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.

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?

## (2)

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 section 2.1 of the textbook (and note errata: `http://csapp.cs.cmu.edu/public/errata.html`), **with the additional restriction** that you may not use equality (==) or inequality (!=) tests.

A. At least one bit of `x` is 0.
B. At least one bit of `x` is 1.
C. Any of the bits in the most significant nibble of `x` is 0.
D. Any of the bits in the least significant nibble of `x` is 1.
E. The most significant and the least significant byte are equal. For **this part** equality (==) testing **is** allowed.
F. Any **even** bit of `x` is 1. The least-significant bit is an even bit, for example. For **this part** you can assume an `int` is 32-bits.

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