Section 3 Worksheet: Integers

Andrew Benson

Conversion Practice

Complete the following table:

Decimal	Binary	Hexadecimal
1	0b1	0x1
21	0b0001_0101	0x15
129	0b1000_0001	0x81
30	0b0001_1110	0x1e

You have the following C code:

```
int x = 5;
```

How would you write C code to convert x into a binary value? Why?
This question doesn't make sense. x is already a binary value, because computers store integer values in binary! In C code, you're allowed to use decimal or hexadecimal literals to refer to those integers (we used 5 here). But regardless of whether we think of the binary or decimal or hexadecimal representations, they're all the same number / value! You can do all your normal arithmetic and bitwise operations on this value without conversions.

Integer Practice

What is the range of an unsigned long? How many bytes does one take up on the Myth machines?
$[0, 2^{64}-1]$, 8 bytes
What is the range of a signed short? How many bytes does one take up on the Myth machines?
$[-2^{15}, 2^{15}-1]$, 2 bytes
Alice, Bob, and Eve are trying to write an expression for the (floored) average of two **unsigned** integers x and y. Alice says to do `(x+y)/2`. Bob says to do `x/2 + y/2`. Eve says to do `x + (y-x)/2`. What do you think about the correctness of these approaches?
Alice's approach could run into overflow issues – try taking the average of 0xFFFFFFFF and 1.
Bob's approach is incorrect due to double integer division – try taking the average of 3 and 5.
Eve's approach is incorrect due to underflow – try x = 5, y = 3 (remember x and y are unsigned!).

Casting Practice

Determine whether each of these evaluates to true or false.

```
-8 < 5 true
```

```
-8 < 5U false

-8U < 5U false (equivalent to the second)

-8 < (signed int) 5U true (equivalent to the first)

-8 < (unsigned int) 5 false (equivalent to the second)

(unsigned char) 256 > 1 false (0 > 1)

(unsigned char) 257 > 1 false (1 > 1)

(unsigned char) 258 > 1 true (2 > 1)
```