Lab 2

2.60 ◆◆

Suppose we number the bytes in a w-bit word from 0 (least significant) to $w/8 - 1$ (most significant). Write code for the following C function, which will return an unsigned value in which byte `i` of argument `x` has been replaced by byte `b`:

```
unsigned replace_byte (unsigned x, int i, unsigned char b);
```

Here are some examples showing how the function should work:

```
replace_byte(0x12345678, 2, 0xAB) --> 0x12AB5678
replace_byte(0x12345678, 0, 0xAB) --> 0x123456AB
```

2.63 ◆◆◆

Fill in code for the following C functions. Function `srl` performs a logical right shift using an arithmetic right shift (given by value `xsra`), followed by other operations not including right shifts or division. Function `sra` performs an arithmetic right shift using a logical right shift (given by value `xsrl`), followed by other operations not including right shifts or division. You may use the computation `8*sizeof(int)` to determine w, the number of bits in data type `int`. The shift amount `k` can range from 0 to $w - 1$.

```
unsigned srl(unsigned x, int k) {
    /* Perform shift arithmetically */
    unsigned xsra = (int) x >> k;
    .
    .
    .

}

int sra(int x, int k) {
    /* Perform shift logically */
    int xsrl = (unsigned) x >> k;
    .
    .
    .

}
```

2.76 ◆◆

Suppose we are given the task of generating code to multiply integer variable `x` by various different constant factors K. To be efficient, we want to use only the operations +, -, and <<. For the following values of K, write C expressions to perform the multiplication using at most three operations per expression.

A. K = 5;
B. K = 9;
C. K = 30;
D. K = -56;

2.83 ◆

Fill in the return value for the following procedure, which tests whether its first argument is less than or equal to its second. Assume the function `f2u` returns an unsigned 32-bit number having the same bit representation as its floating-point argument. You can assume that neither argument is *NaN*. The two flavors of zero, $+0$ and -0, are considered equal.

```
int float_le(float x, float y) {
    unsigned ux = f2u(x);
    unsigned uy = f2u(y);

    /* Get the sign bits */
    unsigned sx = ux >> 31;
    unsigned sy = uy >> 31;

    /* Give an expression using only ux, uy, sx, and sy */
    return ________ ;
}
```