

Homework 2

Two's-Complement Encodings

Assume we have an integer type of 8 bits, fill in the table below.

Value	Two's complement
37	
-15	
	01010101
	10101010

Shift Operation and Type Casting

Consider the following C expressions:

```
short s = -3;
unsigned short us = s;
int i = -52;
unsigned int ui = i;
```

Assume we are running code on **8-bit machine** using two's complement for signed integers. Also assume that right shift of signed values are performed **arithmetically**. A "short" integer is encoded 4 bits. Fill in the table below.

Expression	Binary Representation
us	
ui	
us << 1	
i >> 2	
ui >> 2	
(short) i	
(int) s	

Byte Order

Consider the following function

```
typedef unsigned char * byte_pointer;
void show_bytes(byte_pointer start, int len) {
    int i;
    for (i=0; i<len; i++)
        printf("%.2x", start[i]);
}
```

```
int val = 0x140A0233;  
byte_pointer valp = (byte_pointer) & val;
```

What is the output of the following call to `show_bytes` on big-endian and little-endian machines respectively?

	little-endian	big-endian
<code>show_bytes(valp, 1);</code>		
<code>show_bytes(valp, 2);</code>		
<code>show_bytes(valp, 4);</code>		

Unsigned Addition

Write a function with the following prototype:

```
/* Determine whether arguments can be added without overflow  
 * This function should return 0 if arguments x and y can  
 * be added without causing overflow  
 */  
int uadd_ok(unsigned x, unsigned y);
```