ECE 222 System Programming Concepts
Lecture 3 – Bit masking

In the C language, the smallest data type available is char, which is 1 byte (8 bits):

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

How then can one store a single bit?

One possibility is to use an entire char to store the single bit. This is wasteful.

Another option is to figure out how to manipulate just one bit within a char, or within any data type for that matter. In the C language, this is accomplished using bit manipulators.

&	bitwise AND
|	bitwise OR
>>	right shift
<<	left shift

The bitwise AND operator performs an AND between two values, independently at every bit. For example:

10101100 & 11001010 = 10001000   [line them up vertically – easier to see]

The bitwise OR operator performs an OR between two values, independently at every bit. For example:

10101100 | 11001010 = 11101110   [agaiu, line up vertically]

So what would be the final values of x,y in the following?   [have class work it out]

```
char x,y;
x=17;         /* first put the value into binary; 17 = 00010001 */
y=16;         /* 16 = 00010000 */
x=x&y;        /* x=16 */
y=x|4;        /* y=20 */
```

The bitwise shift operators move the bits left or right the given amount. The left shift adds in new low-order bits, setting them all to zero. The right shift adds in new high-order bits, setting them equal to the value of the original highest order bit. For example:

```
10101100 << 3 = 01100000          10101111 << 3 = 01111000
10101100 >> 3 = 11110101          00101100 >> 3 = 00000101
```

So what would be the final values of x,y in the following?   [have class work it]

```
char x,y;
x=17;           /* first put the value into binary; 17 = 00010001 */
y=16;           /* 16 = 00010000 */
x=x>>3;         /* x=2 */
y=y<<1;         /* y=32 */
```

Bit masking is the art of using bitwise operators to manipulate individual bits within larger data types. Here are a few basic manipulations:

Operation	Formula
Set Nth bit	$x = x \mid 2^N$
Clear Nth bit	$x = x \& (2^T-1-2^N)$, where T=total bits
Read Nth bit	$(x \& 2^N) >> N$

Here are some examples to work in class:

```
char x,y;
x=16;                    /* 00010000 */
x=x|4;                   /* 00010100 */         /* set 3rd bit */
x=x & (255-16);          /* 00000100 */         /* clear 5th bit */
y=(x & 4)>>2;            /* 00000001 */         /* read 3rd bit */
```

[In-class exercise bitset.c]