CS 61C Spring 2010 **TA: Long Wei**
Section 115/6 **Week 2 – Pointers** **cs61c-te@imail.eecs**

Quick Review

N bits represent 2^N things:
How many bits do you need to represent 768 things?
10 bits
Kind men give terminal pets extra zebra yolk:
2^{67} = 128 exbi

With 8 bits, what are the bit patterns for the following? For the last row, what is the decimal value of the given bit pattern?

	Unsigned	Sign & Magnitude	One's Complement	Two's Complement
`-1`	No can do	1000 0001	1111 1110	1111 1111
`MAX`	1111 1111	0111 1111	0111 1111	0111 1111
`MIN`	0000 0000	1111 1111	1000 0000	1000 0000
`0x83`	131	-3	-124	-125

In general, with N bits the max/min for unsigned is 2^N-1/0, and for two's complement the max/min is 2^{N-1}-1/-2^{N-1}.

What are the advantages and disadvantages of each integer representation?
Unsigned can represent about twice as the others in terms of magnitude, but no negatives. =(
S&M (lol) is easier for humans to read, but has two zeroes and the problem of going in the opposite direction after overflow.
One's Complement fixes above flaw, but still has two zeroes.
Two's Complement has one extra negative number, but is otherwise perfect.

Complete the following function `convert()` that takes an unsigned integer as an argument, and returns it's value when interpreted as a sign and magnitude number:

```
int convert(unsigned int signMag){

      return -(signMag >> 31)*(signMag&0x7fffffff);

      /* So the >> right shifts the number's bits by 31 places and leaves
      Only the topmost bit. The & makes the topmost bit 0. We hardcoded the
      31 and the 0x7fffffff mask; later on we'll learn about sizeof and can
      Dynamically adjust to the data size. */
}
```

C details

```
int* p1, p2, p3, p4;
```

Did I just declare four pointers?
No, that would be int *p1, *p2, *p3, *p4. The spacing around the * doesn't matter.

```
if ((5/4) * 100 == 125) printf("C can do math!\n");
```

Did it print?
No, integer division 5/4 is equal to 1, not 1.25. To get correct behavior, cast them or do (5.0/4.0).

CS 61C Spring 2010
Section 115/6

Week 2 – Pointers

TA: Long Wei
cs61c-te@imail.eecs

Pointers

Writing the function swap and complete its call.

```
int foo = 5;
int baz = 42;
swap(&foo, &baz);
printf(“foo is %d, baz is %d\n”, foo, baz);
/* foo is 42, baz is 5 */
```

```
void swap (int *x, int* y) {
        int temp = *x;
        *x = *y;
        *y = temp;
        /* Remember C is pass
        by value! */
}
Alternatively:
```

Slower if compiler sucks→

```
void swap (int *x, int* y) {
        *x ^= *y ^= *x ^= *y;
}
```

What is the output of the following program given this snapshot of memory?

Variable (if any)		a	b	c	p					x	y	
Address	...	171	172	173	174	175	176	177	...	655	656	...
Initial Value		15	19	-5	171	0	255	4		-1	8	

```
int main(int argc, char * argv[]){
        int a = 3, b = 144, c = 170;
        int *p;
        printf(“%d, %d, %d\n”, *p, p, &p);
        p = (int *) foo(a,&c);
        printf(“%d, %d, %d\n”, *p, p, &p);
        bar(&a, &b);
        printf(“%d, %d, %d\n”, a, b, c);
        return 0;
}
```

```
int foo (int x, int * y){
        *y = -12;
        return x + (int) y;
}

void bar (int * x, int * y){
        *x = *y;
        *y = (int) &y;
}
```

3, 171, 174
255, 176, 174
144, 656, -12

Bonus Question

What does this function do?

```
int mystery (unsigned int n)   {
   int count = 8 * sizeof(int) ;
   n ^= (unsigned int) - 1 ;
   while (n)  {
      count-- ;
      n &= (n - 1) ;
   }
   return count ;
}
```

Do your homework 1