CS 61C
Spring 2010

Signed Numbers & C Pointers

Week 2 (1/26)

Scott Beamer (cs61c-ta)
adapted from notes by David Jacobs

Review

- How many bits to represent 768 things?
- 2^{67} =

Encoding Comparison	Definition	Advantages	Disadvantages
Unsigned			
Sign and Magnitude			
One's Complement			
Two's Complement			

Conversion Practice	34_{10}	255_{10}	-104_{10}
Unsigned			
Sign & Magnitude			
One's Complement			
Two's Complement			

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

```
int convert(unsigned int signMag) {
```

Pointer Basics

Pointer Basics	Address Operator (&)	Dereference Operator (*)
`int *x;`	`x = &y;`	`*x = 5;`
Tells the compiler to interpret the variable as an address	Returns the address of the variable provided	Essentially "follows" the pointer to access the referenced data

Write a C function that swaps the value of two variables beyond the function.

Try to determine the output of the following:

```
int main(int argc, char * argv[]){
     int a = 3, b = 5, c = 7;
     int *p;
     printf("%d, %d, %d\n", *p, p, &p);
     p = &a;
     printf("%d, %d, %d\n", *p, p, &p);
     *p = *p + b;
     printf("%d, %d, %d\n", *p, p, &p);
     p = &c;
     printf("%d, %d, %d\n", *p, p, &p);
     *p = a;
     printf("%d, %d, %d\n", a, b, c);
     return 0;
}
```

Assume initially:

- `&a = 72;`
- `&b = 76;`
- `&c = 80;`
- `&d = 92;`

Output: