CSc 352
Bit Manipulation

Benjamin Dicken

Bit Operations

C supports a number of operations to manipulate the ones and zeros in memory

Shifting:	`>>` `<<`
Masking:	`&` `|` `^`
Flipping:	`~`

```c
uint8_t x, y;
x = 1;       // 00000001
y = x<<2;    // 00000100
y = y>>2;    // 00000001

for (int i = 0; i < 8; i++) {
  y = y<<1;
  printf("%u\n", y);
}
```

Activity

Viewing bits on stdout

- Implement the function

```c
void print_bits(uint8_t data);
```

- Should print out the 1s and 0s stored in **data** to standard output
- For example:

```c
uint8_t x = 4;
print_bits(x); // Should print 00000100
```

Activity

Viewing bits on stdout

- Implement the function

```
void print_bits(uint8_t * data, int size);
```

- Should print out the 1s and 0s stored in the array of length **size** that **data** points to
- For example:

```
uint16_t x = 4;
print_bits(x, 2); // Should print 00100000 00000000
```

```c
void print_bits(uint8_t * data, int size){
  uint8_t* copy = malloc(size);
  memcpy(copy, data, size);
  for(int i = 0; i < size; i++){
    for(int j = 0; j < 8; j++){
      uint8_t temp = copy[i];
      temp = temp<<(7-j);
      temp = temp>>7;
      printf("%u", temp != 0 ? 1 : 0);
    }
    printf(" ");
  }
  printf("\n");
  free(copy);
}
```

Permissions

Recall that permissions for files can be represented as a binary sequence:

```
111 110 100
```

Owner can read. write, and execute

Group can read and write

Everyone can read

Permissions

Could represent this with a uint16_t

```
0000000 111 110 100
```

Owner can read. write, and execute

Group can read and write

Everyone can read

Activity

Permissions

- Implement the function

```
uint16_t owner_permissions(uint16_t * permissions);
```

- Should take the Owner permissions and set those same permissions as the group and every permissions too, return the number
- For example:

```
uint16_t x = 272; // 000000000 100 010 000
owner_permissions(x); // Should return 000000000 100 100 100
```