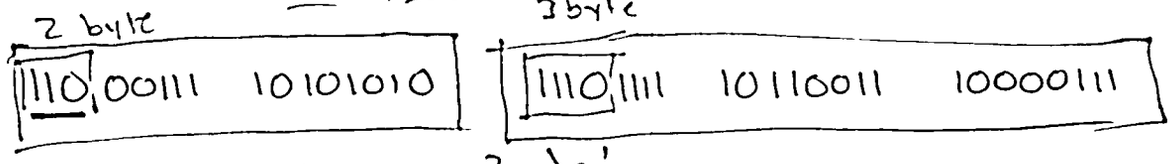
Review Qs:

1: What are the signed and unsigned decimal interpretations of 10000001?

$-128+0+0+0+0+0+0+1 = -127$

$128+0+0+0+0+0+0+1 = 129$

2: How many code points are encoded by this UTF-8 sequence?



3: What does this print?

'a'
97

```
char str[] = {0b01100001, 98, 'c', '\0'};
printf("%s", str);
```

abc

Bitwise Operators

& (and)	| (or)	^ (xor)	~ (not)
0011 & 1010 = 0010	0011 | 1010 = 1011	0011 ^ 1010 = 1001	~1001 = 0110

```
char c1 = 0b01100001;
char c2 = 0b11001110;
char c3 = 0b11100001;
char c4 = 0b11110111;
char mask = 0b11110000;
int is_3byte_utf8(char c) {
    return (c & mask) == 0b11100000;
}
```

c1 & mask = 0110 0000
c2 & mask = 1100 0000
c3 & mask = 1110 0000
c4 & mask = 1111 0000

Shifting operators

<< is *2

>> is /2

<< (left shift)

0b00110001 << 2 = 0b11000100

0b00110001 << 3 = 0b10001000

>> (right shift)

0b00110001 >> 2 = 0b00001100

0b10001100 >> 2

behavior depends on the type of this!

int32_t if value's type is signed: 0b11100011
uint32_t if value's type is unsigned: 0b00100011

0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

16 1

0xA7

A*16 + 7*1

160 + 7

167