```c
char firstfour(char c) {
    return c & 0b11110000;
}
```

```c
char lastfour(char c) {
    return c & 0b00001111;
}
```

```c
char startswith1110(char c) {


}
```

firstfour(0b01010101) = 0b01010000

lastfour(0b11110011) = 0b00000011

firstfour(0b11000000 (192)) = 0b11000000

firstfour(lastfour(x)) = 0

A ✓ return firstfour(c) == 0b11100000

B ✗ return c > 0b11100000

C ✗ return (c & 0b11100000) == 0b11100000

very wrong (Monday)

↑ change to 1 to correct

can you find a counterexample where c doesn't start 1110, but returns true

0b11110000

A 0b11110000 == 0b11100000 (false, correct)

B 0b11110000 > 0b11100000 true (wrong)

C 0b11110000 & 0b11100000 = 0b11100000 == 0b11100000 true (wrong)

```c
#include <stdio.h>
#include <string.h>

void inspect(char s[]) {
    int index = 0;
    printf("%s, length %ld: ", s, strlen(s));
    while(s[index] != 0) {
        char current = s[index];
        printf("%c (%hhu 0b", current, current);
        for(int place = 128; place > 0; place /= 2) {
            if((current & place) == 0) printf("0");
            else printf("1");
        }
        printf(") ");
        index += 1;
    }
    printf("\n");
}

int main() {
    char input[100];
    fgets(input, 100, stdin);
    printf("\n");
    inspect(input);
}
```

single & (bitwise &)

printing the binary rep of a character

```
$ gcc inspect.c -o inspect
$ ./inspect
José, length 5: J (74 0b01001010) o (111 0b01101111) s (115 0b01110011) (195 0b11000011) (169 0b10101001)
$ ./inspect
ピカチュウ
ピカチュウ length 15: (227 0b11100011) (131 0b10000011) (148 0b10010100) // ... continues
$ ./inspect

, length 4: (240 0b11110000) (159 0b10011111) (166 0b10100110) (128 0b10000000)
```

ASCIII does not define é or ピ or . There are 100s of thousands of chars

There are millions of lins of code that use ASCII

1993 USENIX Rob Pike San Diego

UTF-8

Bytes like 0xxxxxxx are ASCII (code points)

Bytes like 1xxxxxxx are part of multi-byte characters (code points)

é is 2 bytes	0b110xxxxx	0b10xxxxxx	code points up to 2^{11}	
ピ is 3 bytes	0b1110xxxx	0b10xxxxxx	0b10xxxxxx	cps up to 2^{16}
is 4 bytes	0b11110xxx	0b10xxxxxx (3 times)		cps up to 2^{21}

"Unicode" is a map from code points to glyph, character