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

void inspect(char s[]) {
    int index = 0;
    printf("%s, length %ld:\n", s, strlen(s));
    while(s[index] != 0) {
        char current = s[index];
        printf(" %c (%03hhu 0b%08hhb)\n", current, current, current);
        index += 1;
    }
    printf("\n");
}

int main() {
    char s1[] = "José";
    char s2[] = "ピカチュウ";
    char s3[] = "😊";
    inspect(s1);
    inspect(s2);
    inspect(s3);
}
```

```
$ gcc inspect.c -o inspect
$ ./inspect
José, length 5:
 J (074 0b01001010)
 o (111 0b01101111)
 s (115 0b01110011)
 Ã (195 0b11000011)
 © (169 0b10101001)

$ ./inspect
ピカチュウ length 15:
 ã (227 0b11100011)
 ƒ (131 0b10000011)
 ª (148 0b10010100)
 // ... continues
$ ./inspect
😊, length 4:
 ð (240 0b11110000)
 Ÿ (159 0b10011111)
 ¦ (166 0b10100110)
 € (128 0b10000000)
```

110 prefix for 2-byte character

10 prefix on "continuation" bytes

1110 prefix for 3-byte character

10 prefix on "continuation" bytes

11110 prefix for 4-byte character

10 prefix on "continuation" bytes

é

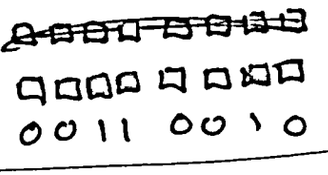
11000011 10101001

00011101001

233 "unicode code point 233"

```
char s[] = "cse29";
printf("%c %d", s[3], s[3]);
```

A1: What does this print? 2 50




```
char s2[] = "cse30_next";
printf("%d", s2[10]);   // fill in to print 0
printf("%c", s2[4]);    // fill in to print 0
```

(A2)

(A3)

		unsigned	signed
0000 0000		0	0
0000 0001		1	1
0000 0010		2	2
0000 0011	ASCII	3	3
⋮		⋮	⋮
0111 1111		127	127
1000 0000		128	-128
1000 0001		129	-127
1000 0010		130	-126
⋮		⋮	⋮
1111 1111		255	-1

'0'iii' 'iiii
+ 0000 0001
10000000

2's complement

most significant digit counts as negative

- addition "just works"
- no wasted/ambiguous reps
- can do "by hand"

UTF8

~~Encoding~~

0xxxxxxx 1-byte UTF8 = ASCII

~~110~~

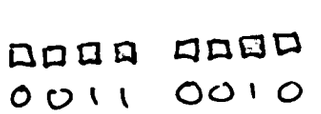
110xxxxx 10yyyyyy 2-byte UTF8 "code point"

xxxxxyyyyyy ← this is a number in the Unicode standard for a "glyph" or character we see

1110xxxx 10yyyyyy 10zzzzzz - 3-byte

11110xxx 10yyyyyy 10zzzzzz 10wwwwww - 4-byte

```
char s[] = "cse29";
printf("%c %d", s[3], s[3]);
```



A1. What does this print? 2 50

```
char s2[] = "cse30 next";
printf("%d", s2[ 10 ]);      // fill in to print ∅
printf("%c", s2[ 4 ]);       // fill in to print ∅
```

A2
A3

	unsigned	signed
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
0000 0011	3	3
0000 0100	4	4
⋮		
0111 1111	127	127
1000 0000	128	-128
1000 0001	129	-127
1000 0010	130	-126
⋮		
1111 1111	255	-1

$$\begin{array}{r} 0111\ 1111 \\ +\quad\quad\ 1 \\ \hline 10000000 \end{array}$$

2's complement

Idea: most significant bit is interpreted as <u>negative</u>

- addition "just works"
- unique/unambiguous 0
- adding leading 1's to a negative # does not change its meaning

UTF-8

1-byte	0xxxxxxx	ASCII		
2-byte	110xxxxx	10yyyyyy		xxxxxyyyyyy → this is binary # of a code point in Unicode
3-byte	1110xxxx	10yyyyyy	10zzzzzz	
4-byte	11110xxx	10yyyyyy	10zzzzzz	10wwwwww