Announcements

Test next week Mon - Sat

Like the quizzes

Test 1: 30 min

Test 2+3: 45 min 1pm today re-do res.

Ctrl-e to see tabs

Ctrl-Shift-V to paste

PA1 due last night

0, 1, 2, 3 grade

↳ good but we have feedback

0, 1 → 2

2 → 3

resubmission improvement

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

void print_as_bin(char c) {
    for(int place = 128; place > 0; place /= 2) {
        if((c & place) == 0) printf("0");
        else printf("1");
    }
}

int main() {
    char s = 200;
    unsigned char u = 200;

    printf("s: ");
    print_as_bin(s);
    printf("\t\tu: ");
    print_as_bin(u);
    printf("\n");

    printf("s as hhx: %hhx\t\tu as hhx: %hhx\n", s, u);
    printf("s as x: %x\tu as x: %x\n", s, u);

    printf("s < 127: %d\t\tu < 127: %d\n", s > 127, u > 127);
}
```

hhx — use the width/size of the argument

x

```
$ gcc signed_hex.c -o signed_hex
$ ./signed_hex
s: 11001000             u: 11001000
s as hhx: c8            u as hhx: c8
s as x: ffffffc8        u as x: c8
s < 127: 0              u < 127: 1
```

wtf — "sign-extending" printf "wants" to print 4 bytes for %x

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

int32_t code_point2(char c1, char c2) {
    char part1 = c1 & 0b00011111;
    char part2 = c2 & 0b00111111;
    return (part1 << 6) + part2;
}

int32_t code_point3(char c1, char c2, char c3) {
    char part1 = c1 & 0b00001111;
    char part2 = c2 & 0b00111111;
    char part3 = c3 & 0b00111111;
    return (part1 << 12) + (part2 << 6) + part3;
}

int main() {
    char s[] = "é";
    printf("%d\n", code_point2(s[0], s[1]));

    char s2[] = "□";
    printf("%d\n", code_point3(s2[0], s2[1], s2[2]));
}
```

x format char stands for

hexadecimal → base 16

decimal → base 10

binary → base 2

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

2 hex digits represent 1 byte

1100 1000

C 8

UTF-8 prefixes in hex

1 byte	0-7
2 byte	C or D
3 byte	E
4 byte	F

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

void capitalize(char s[]) {
    uint32_t index = 0;
    printf("sizeof(s) = %ld\tstrlen(s) = %ld\ts: %p\n", sizeof(s), strlen(s), s);
    while(s[index] != 0) {
        if(s[index] >= 'a' && s[index] <= 'z') {
            s[index] -= 32;
        }
        index += 1;
    }
}

int main() {
    char h[] = "hello";
    printf("sizeof(h) = %ld\tstrlen(h) = %ld\th: %p\n", sizeof(h), strlen(h), h);
    capitalize(h);

    printf("%s\n", h);


    char g[] = "greetings i'm really excited to be here";
    printf("sizeof(g) = %ld\tstrlen(g) = %ld\tg: %p\n", sizeof(g), strlen(g), g);
    capitalize(g);
    printf("%s\n", g);
}
```

```
$ gcc sizeof_not_strlen.c
In function ‘’capitalize:
7:63: warning: ‘’sizeof on array function parameter ‘’s will return size of ‘char ’* [-Wsizeof-array-argument]
    7 |     printf("sizeof(s) = %ld\tstrlen(s) = %ld\ts: %p\n", sizeof(s), strlen(s), s);
      |                                                               ^
5:22: note: declared here
    5 | void capitalize(char s[]) {
      |                 ~~~~~^~~
$ ./a.out
sizeof(h) = 6   strlen(h) = 5   h: 0x7ffff284638a
sizeof(s) = 8   strlen(s) = 5   s: 0x7ffff284638a
HELLO
sizeof(g) = 40  strlen(g) = 39  g: 0x7ffff2846390
sizeof(s) = 8   strlen(s) = 39  s: 0x7ffff2846390
GREETINGS I'M REALLY EXCITED TO BE HERE
```