```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);
}
```

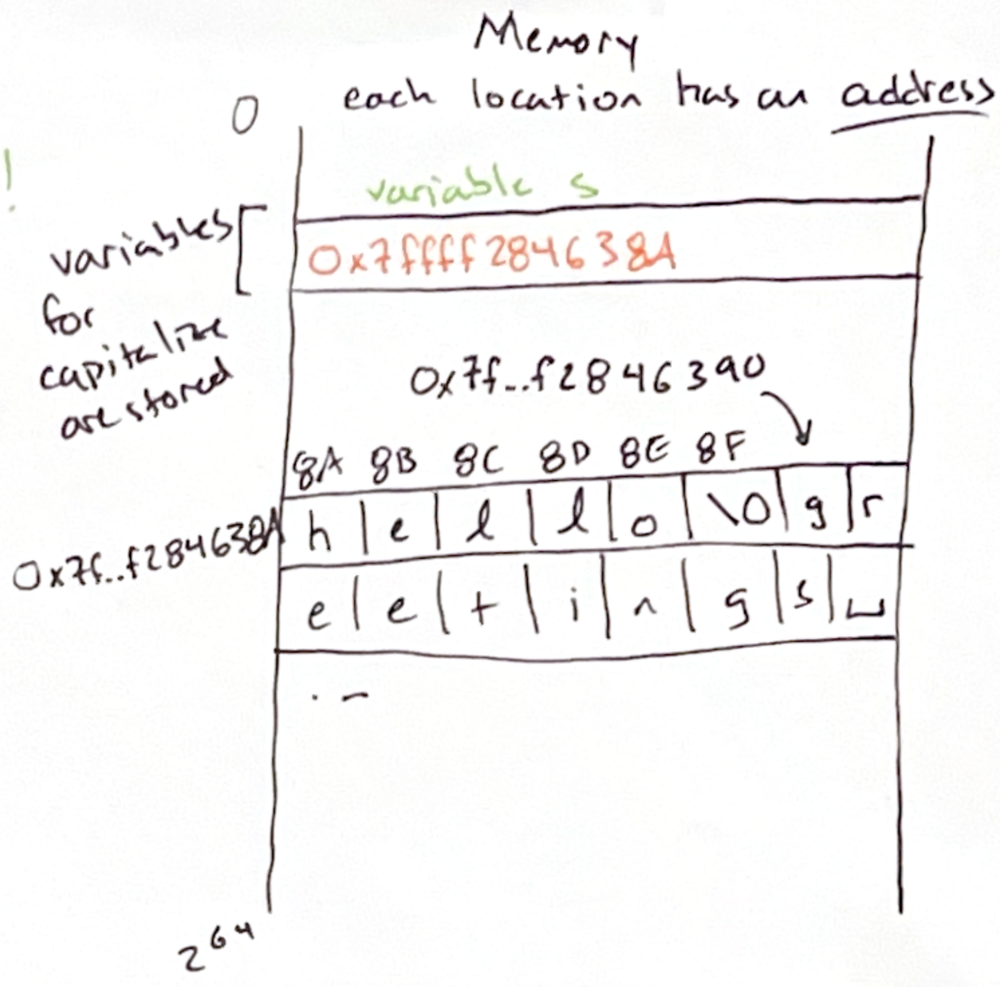
%p: "pointer" means "address"

no size information! and it is diffrent at diffrent times!

look up/assign value offset from address in s

h's type char[6]

g's type is char[40]




```
$ 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
```

h is size 6

g is size 40

sizeof from type

addresses

addresses

8a + 6 = 90

8*16 + A + 6 = 90

8*16 + 10 + 6 = 90

Why sizeof(s) == 8

Addresses are 8 bytes long (2^{64})

Questions / notice

What's the 0x7ffff business?

sizeof(s) seems to always return 8

h vs. s printed with %p
are the same
(also g vs. s)

h vs. s / g vs. s have the same
strlen result