Pointers and arrays are closely related

Say we have array `int a[]`.

- `a[0]` and `*a` are equivalent
- ` [...] ` is a combination of dereferencing and pointer addition
  - `*(a + 3)` is a synonym for `a[3]`
  - `(a + 3)` is a synonym for `&a[3]`
Pointers & arrays

You’ll notice the differences between arrays and pointers when using sizeof

```c
#include <stdio.h>
int main() {
    int a[] = {0, 1, 2, 3, 4, 5};
    int *a_ptr = a;
    printf("sizeof(a)=%d, sizeof(a_ptr)=%d\n", 
           (int)sizeof(a), (int)sizeof(a_ptr));
    return 0;
}
```

$ gcc -c ptr_sizeof_eg1.c -std=c99 -pedantic -Wall -Wextra
$ gcc -o ptr_sizeof_eg1 ptr_sizeof_eg1.o
$ ./ptr_sizeof_eg1
sizeof(a)=24, sizeof(a_ptr)=8
Passing array as argument *converts it to a pointer*, losing any information about how long it is

- Sometimes called “array decaying”
```c
#include <stdio.h>

void f1(int arg[10]) { printf("f1: %lu\n", sizeof(arg)); }  
void f2(int arg[])  { printf("f2: %lu\n", sizeof(arg)); }     
void f3(int *arg)   { printf("f3: %lu\n", sizeof(arg)); }     

int main() {  
    int one_thru_ten[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
    printf("main: %lu\n", sizeof(one_thru_ten));  
    f1(one_thru_ten);  
    f2(one_thru_ten);  
    f3(one_thru_ten);  
    return 0;  
}
```
$ gcc -o decay1 decay1.c -std=c99 -pedantic -Wall -Wextra

decay1.c: In function 'f1':
decay1.c:3:50: warning: 'sizeof' on array function parameter 'arg' will return
size of 'int *' [-Wsizeof-array-argument]
   void f1(int arg[10]) { printf("f1: %lu\n", sizeof(arg)); }
      ^
decay1.c:3:13: note: declared here
   void f1(int arg[10]) { printf("f1: %lu\n", sizeof(arg)); }
      ^~~
decay1.c: In function 'f2':
decay1.c:4:50: warning: 'sizeof' on array function parameter 'arg' will return
size of 'int *' [-Wsizeof-array-argument]
   void f2(int arg[]) { printf("f2: %lu\n", sizeof(arg)); }
      ^
decay1.c:4:13: note: declared here
   void f2(int arg[]) { printf("f2: %lu\n", sizeof(arg)); }
      ^~~

$ ./decay1
main: 40
f1: 8
f2: 8
f3: 8

Compiler warns you
Pointers & arrays

This fits with what we know

- Passing an array is “pass by pointer,” since arrays decay into pointers when passed
- This is also why we can modify an array in the callee and see the changes in the caller