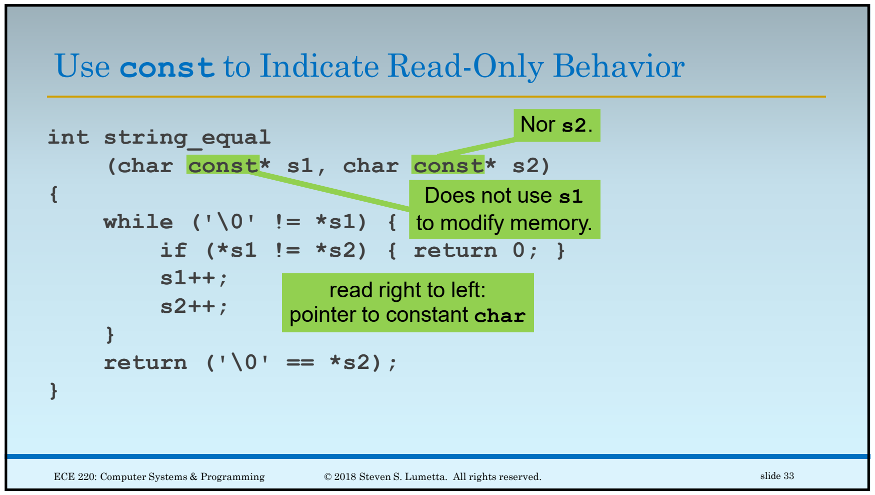


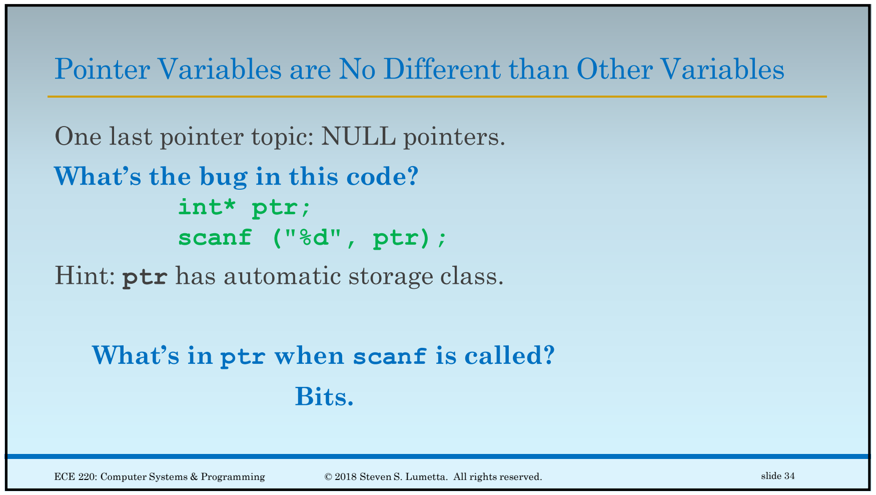
Use `const` to Indicate Read-Only Behavior

```
int string_equal
    (char const* s1, char const* s2)
{
    while ('\0' != *s1) {
        if (*s1 != *s2) { return 0; }
        s1++;
        s2++;
    }
    return ('\0' == *s2);
}
```

Nor **s2**.

Does not use **s1** to modify memory.

read right to left: pointer to constant **char**



Pointer Variables are No Different than Other Variables

One last pointer topic: NULL pointers.

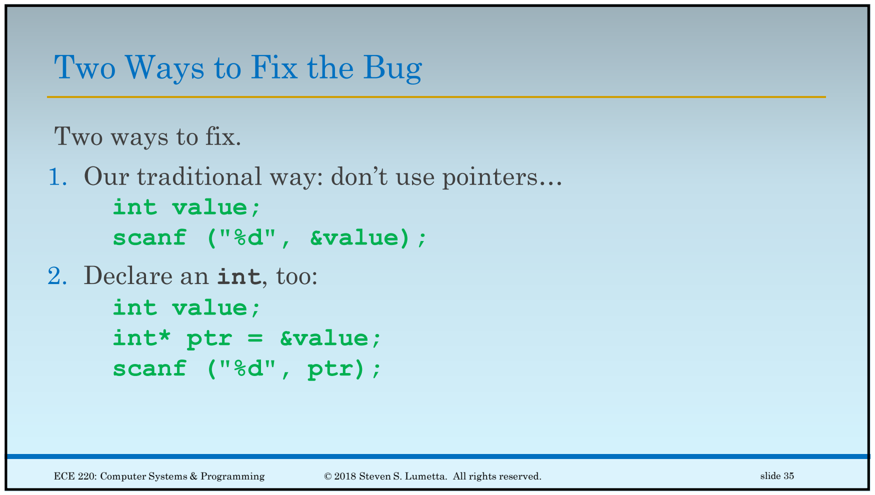
What's the bug in this code?

```
int* ptr;
scanf ("%d", ptr);
```

Hint: **ptr** has automatic storage class.

What's in `ptr` when `scanf` is called?

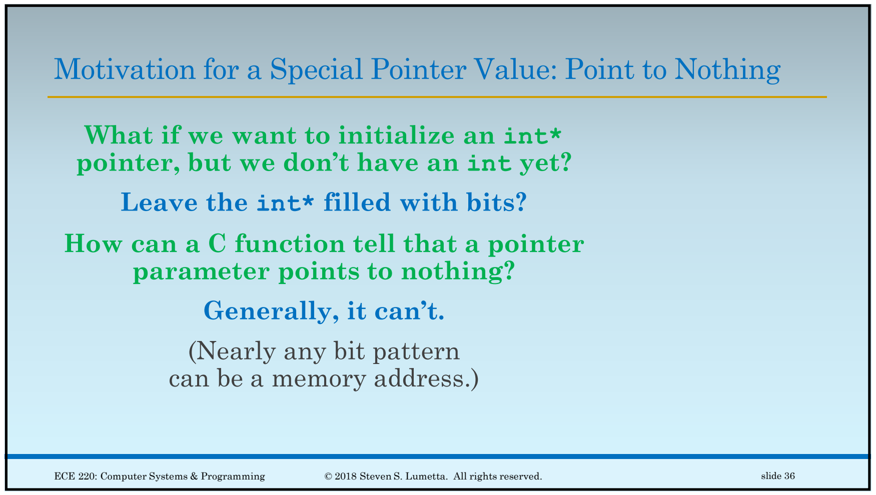
Bits.



Two Ways to Fix the Bug

Two ways to fix.

1. Our traditional way: don't use pointers…
   ```
   int value;
   scanf ("%d", &value);
   ```
2. Declare an **int**, too:
   ```
   int value;
   int* ptr = &value;
   scanf ("%d", ptr);
   ```



Motivation for a Special Pointer Value: Point to Nothing

What if we want to initialize an `int*` pointer, but we don't have an `int` yet?

Leave the `int*` filled with bits?

How can a C function tell that a pointer parameter points to nothing?

Generally, it can't.

(Nearly any bit pattern can be a memory address.)