Never return a pointer to an *automatic* local variable:

```
int *f(void)
{
  int i;
  ...
  return &i;
}
```

The variable `i` doesn't exist once `f` returns, so the pointer to it will be invalid. Some compilers issue a warning such as "*function returns address of local variable*" in this situation.

Pointers can point to array elements, not just ordinary variables. If `a` is an array, then `&a[i]` is a pointer to element `i` of `a`. When a function has an array argument, it's sometimes useful for the function to return a pointer to one of the elements in the array. For example, the following function returns a pointer to the middle element of the array `a`, assuming that `a` has `n` elements:

```
int *find_middle(int a[], int n) {
  return &a[n/2];
}
```

Chapter 12 explores the relationship between pointers and arrays in considerable detail.

Q & A

*Q: **Is a pointer always the same as an address? [p. 242]**

A: Usually, but not always. Consider a computer whose main memory is divided into ***words*** rather than bytes. A word might contain 36 bits, 60 bits, or some other number of bits. If we assume 36-bit words, memory will have the following appearance:

Address	Contents
0	001010011001010011001010011001010011
1	001110101001110101001110101001110101
2	001110011001110011001110011001110011
3	001100001001100001001100001001100001
4	001101110001101110001101110001101110
	⋮
n-1	001000011001000011001000011001000011