Pointers and const

Using const with pointers has some subtle aspects (pointers always seem to have subtle aspects), so let's take a closer look. You can use the const keyword two different ways with pointers. The first way is to make a pointer point to a constant object, and that prevents you from using the pointer to change the pointed-to value. The second way is to make the pointer itself constant, and that prevents you from changing where the pointer points. Now for the details.

First, let's declare a pointer pt that points to a constant:

```
int age = 39;
const int * pt = &age;
```

This declaration states that pt points to a const int (39, in this case). Therefore, you can't use pt to change that value. In other words, the value *pt is const and cannot be modified:

```
*pt += 1;          // INVALID because pt points to a const int
cin >> *pt;        // INVALID for the same reason
```

Now for a subtle point. Our declaration for pt doesn't necessarily mean that the value it points to is really a constant; it just means the value is a constant insofar as pt is concerned. For example, pt points to age, and age is not const. You can change the value of age directly by using the age variable, but you can't change the value indirectly via the pt pointer:

```
*pt = 20;         // INVALID because pt points to a const int
age = 20;         // VALID because age is not declared to be const
```

In the past, we've assigned the address of a regular variable to a regular pointer. Now we've assigned the address of a regular variable to a pointer-to-const. That leaves two other possibilities: assigning the address of a const variable to a pointer-to-const and assigning the address of a const to a regular pointer. Are they both possible? The first is, and the second isn't:

```
const float g_earth = 9.80;
const float * pe = &g_earth;  // VALID
```