A Note About Implicit Constructors

The subtleties of implicit conversions are explained here. The rule to remember is that C++ will only perform implicit conversions with a one-argument constructor. More accurately, it will perform the conversion if the class has a constructor that can be called with one argument. For example, consider the class definition:

```
class MyClass
{
public:
        MyClass(int a=0, int b = 0): x(a), y(b) {}
private:
        int x;
        int y;
};
```

The following is a legal implicit conversion of an integer to an object of type MyClass:

```
MyClass  A = 6;   // converts 6 into the object with x=6, y = 0 and copies it into A
```

but

```
MyClass  B = {4,5};
```

is illegal because it attempts to convert an aggregate structure into a class object.

Note that it makes no difference whether the object is a class or a struct. If I replace the definition with

```
struct MyClass
{
public:
        MyClass(int a=0, int b = 0): x(a), y(b) {}
private:
        int x;
        int y;
};
```

the same conversions are allowed.