**ECE198KL: Introduction to Computer Engineering II** **Spring 2013**
**Notes**

## Additional Notes on References in C++

As a supplement to the lectures and the notes already put online, I want to give you a side-by-side comparison of code using references and using pointers.

The class below wraps up an integer simply for the purpose of our example.

```
class ALPHA {
    private:
        int num;
    public:
        ALPHA (int val) :  num (val) { }
        int getNum () const { return num; }
        void setNumFromPtr (int* where) { num = *where; }
};
```

Now let's see how code written using pointers compares with code written using references. On the left below is a simple function written with pointer arguments. On the right below is the same function (slightly different name) written using references. Below each function is an example call. The variable `a` has type `ALPHA`, while the variables `one` and `two` are `int`s.

Since a reference in C++ is implemented as a pointer, *the assembly code generated in both cases is identical.* The syntax used for references is different, however: a reference of type `SomeClass&` acts like an object of type `SomeClass` in the code.

Note that we do not encourage you to use non-constant references. The example is offered only as an illustration of how references work.

```
bool compareXchg (ALPHA* alpha,
     int* compare, int* newVal)
{
    if (alpha->getNum () == *compare) {
        alpha->setNumFromPtr (newVal);
        return true;
    }
    return false;
}

compareXchg (&a, &one, &two);
```

```
bool compareXchgRef (ALPHA& alpha,
     int& compare, int& newVal)
{
    if (alpha.getNum () == compare) {
        alpha.setNumFromPtr (&newVal);
        return true;
    }
    return false;
}

compareXchgRef (a, one, two);
```

This code is also available to you in the `example.cc` file.