# ECE220
# Honors Lab Section

Lab 10: C++ Access, Inheritance, and Templates

# Classes

- Can be constructed in two ways:
  - using class notation
    - defaults to private access
  - using struct notation
    - default to public access
    - backwards compatible with C (to a degree)
- Usually split between a header and a .cpp file
  - Though can have everything in the header

# Classes

```
class Point {
        int x, y;

        public:
                Point(int x_in, int y_in) : x(x_in), y(y_in) { }
                ~Point();
                void set_point(int x_in, int y_in) {
                        x = x_in; y = y_in;
                }
                void get_point(int *x_out, int *y_out) {
                        *x_out = x; *y_out = y;
                }
};
```

# Classes

```
struct Point {
        Point(int x_in, int y_in) : x(x_in), y(y_in) { }
        ~Point();
        void set_point(int x_in, int y_in) {
                x = x_in; y = y_in;
        }
        void get_point(int *x_out, int *y_out) {
                *x_out = x; *y_out = y;
        }

        private:
                int x, y;
};
```

# Constructor/Destructor

- Constructor
  - Called whenever we create a new object
  - Initialization of variables and etc.
  - Initializer list
- Destructor
  - Called whenever the object goes out of scope or call delete on it
  - Releases resources taken up by object
- Defaults
  - Compiler generates default constructors/destructors

# Access Specifiers

| Public | | | |
|---|---|---|---|
| **Specifier** | **Derived access specifier** | **Derived class access** | **Public access** |
| Public | Public | Yes | Yes |
| Protected | Protected | Yes | No |
| Private | Private | No | No |

| Protected | | | |
|---|---|---|---|
| **Specifier** | **Derived access specifier** | **Derived class access** | **Public access** |
| Public | Protected | Yes | No |
| Protected | Protected | Yes | No |
| Private | Private | No | No |

| Public | | | |
|---|---|---|---|
| **Specifier** | **Derived access specifier** | **Derived class access** | **Public access** |
| Public | Private | Yes | Yes |
| Protected | Private | Yes | No |
| Private | Private | No | No |

# Function Templates

- Allows for generic programming

```
template<typename T>
T max(T const &a, T const &b){
        return (a < b) ? b : a;
}
```

- Calling

```
int i_val = max(1, 2);
double d_val = max(1.1, 2.2);
char c_val = max('a', 'b')
```

# Class Templates

- Apply templates to classes

```
template<typename T>
class Point {
        T x, y;

        public:
                Point(Tx_in, T y_in) : x(x_in), y(y_in) { }
                ~Point();
                void set_point(T x_in, T y_in) {
                        x = x_in; y = y_in;
                }
                void get_point(T *x_out, T *y_out) {
                        *x_out = x; *y_out = y;
                }
};
```

# Class templates

- Although implementing a templated class is similar to a regular class, you can't simply create a separate .cpp file for it
  - Need to include implementation in the header file
  - Create a separate file with implementation and that file in the header file
  - Lab 10 uses a .tpp (templates++) file to house the implementation code

- Lab 10 demo and concepts application.