CS 3460

Introduction to Type Aliases

Type Aliases

- Ability to create an alias for a data type
  - `typedef` (legacy)
  - `using` (new C++ goodness)
- What is the difference between the two?
  - Nothing
  - But, with C++ 11 standard `using` is the preferred approach to match other assignment patterns

typedef

- Form: `typedef [type] [alias];`
- Where
  - `typedef` : required keyword
  - `[type]` : existing type
  - `[alias]` : alias to use for the type

```
typedef int MyInt;
typedef std::string MyString;

MyInt a = 10;
MyString s = "Hi Mom!";
```

using

- Form: `using [alias] = [type];`
- Where
  - `using` : required keyword
  - `[type]` : existing type
  - `[alias]` : alias to use for the type

```
using MyInt = int;
using MyString = std::string;

MyInt a = 10;
MyString s = "Hi Mom!";
```

using – One More Thing

- There is one thing using can do typedef can't
  - Will make more sense with generic programming

```
template <typename T>
class Original {};

template <typename T>
using Alias = Original<T>;
```

- `Alias<T>` is a type alias for `Original<T>`