Reminders

- HW2 due Thursday, 26th Oct
- Midterm on Friday, 3rd Nov
- Review section, Thursday, 2nd
This or that?

- Consider the following code:

**Pointers:**

```c
int i;
int *pi = &i;
```

**References:**

```c
int i;
intr &ri = i;
```

In both cases,

The difference lies in how they are used in expressions:

```c
*pi = 4;
ri = 4;
```
References Example

// Part 1
int i = 0, j = 4;
int *pi = &i;

// Part 2
int &ri = i;

// Part 3
*pi = 3;

// Part 4
ri = j;
Pointers and References

• Once a reference is created, it cannot be later made to reference another object.
  • Compare to pointers, which are often reassigned.
• References can’t be initialized to null, whereas pointers can.
• References can never be uninitialized. It is also impossible to reinitialize a reference.
• Demo: experiments.cc
C++ const declaration

• As a declaration specifier, const is a type specifier that makes objects unmodifiable.
  
  ```cpp
  const int m = 255;
  ```

• Reference to constant integer:
  
  ```cpp
  int n = 100;
  const int &ri = n;  // ri becomes read only
  ```

• Demo: const.cc
When to use?

- **Pointers**: may point to many different objects during its lifetime. Pointer arithmetic (++ or --) enables moving from one address to another. (Arrays, for e.g.)
- **References**: can refer to only one object during its lifetime.
- **Style Guide Tip**:  
  - use const reference parameters to pass input  
  - use pointers to pass output parameters  
  - input parameters first, then output parameters last
/* Note: This code is unfinished! Beware! */
class Point {

public:
    Point(const int x, const int y); // constructor
    int get_x() const { return x_; } // inline member function
    int get_y() const { return y_; } // inline member function
    double distance(const Point &p) const; // member function
    void setLocation(const int x, const int y); // member function

private:
    int x_; // data member
    int y_; // data member
}; // class Point
Complex example

- Code Review and Demo: complex_example (lec11-code)
- Note the friend functions
- Friend functions are
  - NOT member functions
  - declared within a class definition with keyword friend
  - have the right to access private and protected members of the class
Section Exercise

- Define a class Rectangle whose instance variables are a pair of Point objects (upper left, lower right).
- Include at least one constructor. Make sure you get const right in the right places.
- Methods:
  - `getul()`, `getlr()` - returns upper and lower points.
  - `intersect(Rectangle &r)` – returns a Rectangle representing the overlap.
  - `area()` - returns the Rectangle's area.
  - `contains(Point &p)` - returns true or false depending on whether point p is inside the rectangle.
- The C++ Primer text and cplusplus.com contain good reference material.