CSE 333 – SECTION 4

References, Classes and const.
This or that?

- Consider the following code:

**Pointers:**

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

**References:**

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

In both cases,

The difference lies in how they are used in expressions:

```c
*pi = 4;
ri = 4;
```
C++ const declaration

- As a declaration specifier, `const` is a type specifier that makes objects unmodifiable.

```cpp
int const m = 255;
```

- Reference to constant integer:

```cpp
int n = 100;
int const &ri = n;  // ri becomes read only
```
When to use?

- Function parameter types and return types and functions that declare overloaded operators.

- **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
void swap(int &x, int &y) {
    int tmp = x;
    x = y;
    y = tmp;
}

int main(int argc, char **argv) {
    int a = 5, b = 10;
    swap(a, b);
    cout << "a: " << a << "; b: " << b << endl;
    return EXIT_SUCCESS;
}
C++ Classes

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
Section Exercise – Part I (Required)
Due Friday (7/13) by 11pm

- 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.
  - `cornerPoints()` – to obtain the corner points.
  - `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.
Part II (Optional)

- Add a second constructor that takes 4 coordinates (minx, maxx, miny, maxy) and creates the upper left/lower right Point instance variables.
- Make the first constructor smart enough so the points can be any two diagonal corners and the constructor figures out what the top/bottom/left/right coordinates are and constructs upper left/lower right instance Point instance variables accordingly.
- Additional Methods:
  - `Intersects(Rectangle &other)` - returns true if this rectangle intersects the other one.
  - `BoundingBox(Rectangle &other)` - returns a new rectangle that tightly encloses both this rectangle and other.