Week 8

kstatz@colum.edu

The Problem With pointers

Raw pointers are the basic way of returning something or passing something by reference. Pointers are used when the life cycle of a variable in memory is longer than the function that declared it. The problem is ownership. Who owns the pointer once it has been returned from a method?

Returning By Value

```
T foo() {
  T t = T();
  return t;
}
```

This code block allocates t on the stack and, worst case, will copy t to the caller. Copying happens because we cannot share stack memory outside of the function scope.

Named Return Value optimization

NRVO is a compiler optimization for return values. When the compiler sees the following block:

```
struct A {
  A();
  A(const A &) { std::cout << "Copy" << std::endl; }
};

A foo() { return A(); }


std::cout << "Hello World" << std::endl;
A a = foo();
```

it will either print out one of the following based on compiler settings, version etc

"Hello World" "Copy" "Copy"

This is worst case where the constructor call to A() gets copied to a hidden temporary value and then copied t the variable 'a' on return

"Hello World" "Copy"

This is generally what happens without optimization. the value generated by the constructor call gets copied to 'a'

"Hello World"

This is with NRVO, where the function return is replaced by the value which is A();

Optimizing value returns with move semantics

starting in c++11 move operations. Move "steals" the value from a variable and moves it to another skipping the copy where both values exist for a time.

```
T foo() {
  T t = T();
  std::move(t);
  return t;
}
```

Shared Pointers