CS 162
Intro to CS II
Classes
Odds and Ends...

• Last week to demo Assignment #1!!!
• Continue to work on Assignment 2
• Assignment 2 questions???
Let’s revisit our library...

• Where do we need the Big Three?
  – Only in cart and library
  – The implicit Big Three is okay in classes without dynamic memory
The Big “Three”

• If dynamic memory allocation in class, then...
  – Destructor
  – Copy Constructor
  – Assignment operator overload
What is a Destructor?

• Deallocate any member variable dynamically allocated...

• What would this destructor look like then?

```cpp
string::~string() {
    delete [] s; //delete ignores NULL
}
```
What is a copy constructor?

• Used in pass by value
• Returning an object from a function
• Pass the class type to a constructor

//The copy constructor has to have parameter (const same_class_type &)
string:: string(const string &other) {
    len=other.len;
    if(len == 0) s=NULL;
    else {
        s=new char[len];
        for(int i=0; i<len; i++)
            s[i] = other.s[i];
    }
}
What is assignment overload?

//Looks like copy constructor but has void return type and delete
void string::operator=(const string &other) {
    if(s!=NULL) delete [] s; //have to delete before copying over
    len=other.length();
    if(len == 0) s=NULL;
    else {
        s=new char[len];
        for(int i=0; i<len; i++)
            s[i] = other.at(i);
    }
}
```cpp
#include "library.h"
#include <iostream>
using namespace std;

//copy constructor used to pass object by value, and destructor called when object goes out of scope
void fun(library l) {
    cout << l.get_patron()->get_name() << endl;
}

//copy constructor used to return object by value, and destructor called when object goes out of scope
library fun() {
    library lib;
    return lib;
}

int main() {
    library l;
    library l2=fun(); //copy constructor called to create l2, same as below
    //library l2(fun()); //copy constructor called to create l2

    //assignment operator overload called to copy returned object from fun to l2, since l2 has already been created!!!
    l2=fun();
    cout << l.get_patron()->get_name() << endl;
    l.get_patron()->set_name("jennifer");
    cout << l.get_patron()->get_name() << endl;
    fun(l); //call the copy constructor to pass by value
    return 0;
}
```
```cpp
#include "patron.h"

class library {
    private:
        //direct initialization of object in class only allowed in 2011 standard and above, and it doesn't have the patron p("jen") //format as it does outside the class as a non-member //patron p=patron("jen");
    patron p;

    public:
        library(); //default constructor
        library(const library &); //copy constructor
        void operator=(const library &); //assignment operator overload
        ~library(); //destructor
        patron * get_paren();
        void set_paren(const patron &);
};
```
```cpp
#include "library.h"
#include <iostream>
using namespace std;

// need to call patron's non-default constructor before
// the curly brace because that is when the object is
// created
library::library() : p("jen") {
    // can't do this because p would have already been created
    // with the default constructor, & we don't have a default
    // p=patron("jen");
}

// copy constructor takes argument of same type as the object being
// constructed and it must be a reference
library::library(const library &l) {
  library::library(const library &l) : p(l.p.get_name()) {
    // p=l.p; // assignment overload in patron called
    // p.set_name(l.p.get_name()); // use accessors/mutators to set member
    cout << "Library object copy constructor!" << endl;
  }

  // assignment operator overload looks like copy constructor but with void
  // return type and the object and members have already been constructed!
  void library::operator=(const library &l) {
    // library::library(const library &l) {
    p=l.p; // assignment overload in patron called
    // p.set_name(l.p.get_name());
    cout << "Library object assignment overload!" << endl;
  }

  // destructor created with ~ in front of class name
  library::~library() {
    cout << "Library object being destructed!" << endl;
  }
```