CS 162
Intro to CS II
Finish Inheritance/Polymorphism
Odds and Ends...

• Assignment #3 due Sunday at midnight!
Make Destructors Virtual

• What does this do if destructor isn’t virtual?
  Parent *p = new Child;
  ...
  delete p;

• **Example:**
  class parent {
    public:
      parent() {  //Have a constructor
        shared_ptr = new int;
      }
      virtual ~parent() {  //Have a destructor
        delete shared_ptr;
      }
    private:
      int *shared_ptr;
  }
Adding an owner and a Car vin...

• What if we added an owner to vehicles?
• What if we added a car class with a vin?
• How do we convince ourself the destructor needs to be virtual?

• How would we make vehicle an abstract class?

• How can you make your assignment 3 polymorphic?
Demo...
Demo...

```cpp
#include "vehicle.h"
#include <cstdlib>
#include <cstring>
#include <iostream>
using namespace std;

vehicle::vehicle(int seats) {
    this->seats = seats;
    owner = new char[strlen("jen") + 1];
    strcpy(owner, "jen");
}

vehicle::~vehicle() {
    cout << "vehicle destruct" << endl;
    delete [] owner;
}

int vehicle::get_seats() {
    return seats;
}

// We made it a pure virtual function
/*int vehicle::get_toll() {
    return 10 * seats;
}*/
```
#pragma once

class vehicle {
    protected:
    int seats;
    char *owner;
    //int toll amount;

class public:
    vehicle(int);
    virtual ~vehicle();
    int get_seats();
    virtual int get_toll() = 0;
};
```cpp
#include "vehicle.h"

class car: public vehicle {
  private:
    char *vin;
  public:
    car(int);
    ~car();
    int get_toll();
};
```
```cpp
#include "car.h"
#include <iostream>
#include <cstring>
using namespace std;

car::car(int s) : vehicle(s) {
    vin=new char[strlen("123")]+1;
    strcpy(vin,"123");
}

car::~car() {
    cout << "car destruct" << endl;
    delete [] vin;
}

int car::get_toll() {
    return 5*seats;
}
```
```cpp
#include "car.h"
#include <iostream>
using namespace std;

int main() {
    vehicle v(4), *vptr;
    bike b(2);
    car c(3);

    // vptr=&c; //not going to show destructor polymorphism
    vptr=new car(5);
    cout << c.get_seats() << endl;
    cout << c.get_toll() << endl;

    cout << vptr->get_seats() << endl;
    cout << vptr->get_toll() << endl;
    delete vptr;

    // cout << v.get_seats() << endl;
    // cout << v.get_toll() << endl;
    cout << b.get_seats() << endl;
    cout << b.get_toll() << endl;
}
```