CS 162
Intro to CS II

More Big Three Concepts
assignment operator overload...

```cpp
14 date::date(const date &other) : m(other.day) {
15     cout << "in the copy constructor" << endl;
16     //month=other.month; //shallow copy
17     month=new char[strlen(other.month)+1];
18     strcpy(month, other.month);
19     day=other.day;
20     year=other.year;
21 }
22 date::~date() {
23     cout << "in the destructor" << endl;
24     delete [] month; //delete memory when obj goes out of scope
25 }
26 //what needs to be in the assignment overload? Does it look just
27 //like the copy constructor? Do you need to delete memory?
28 void date::operator=(const date &other) {
29     cout << "in the operator =" << endl;
30     //month=other.month; //no shallow copy
31     delete [] month;
32     month=new char[strlen(other.month)+1];
33     strcpy(month, other.month);
34     day=other.day;
35     year=other.year;
36 }
```
.h Safeguard for game class...

```c
#include <stdio.h>

struct card {
    int value;
    char suit;
};

// you can use this instead of ifdef directive
#define CARD_HEADER  // if the CARD_HEADER macro hasn't been defined
#define CARD_HEADER  // then define the macro and include the struct

```

OSU Oregon State University
dealer.h includes card.h

```cpp
#include "card.h"

class deck {
    card cards[52];
    public:
    deck();
};
```
deck.cpp includes deck.h

```cpp
#include "deck.h"
#include <iostream>

using namespace std;

deck::deck()
{
    cout << "in deck default const" << endl;
}
```
hand.h includes card.h

```cpp
#include "card.h"

class hand {
    card h[4];
    public:
        hand();
        hand(int);
        ~hand();
        hand(const hand &);
        void operator =(const hand &);
};
```
hand.cpp includes hand.h

```cpp
#include "hand.h"
#include <iostream>

using namespace std;

hand::hand()
    cout << "in hand default const" << endl;

hand::hand(int b)
    cout << "in hand non-default const" << endl;

hand::hand(const hand &other)
    cout << "in hand copy const" << endl;

hand::~hand()
    cout << "in hand destruct" << endl;

void hand::operator=(const hand &other)
    cout << "in hand assignment op overload" << endl;
```

"hand.cpp" 21L, 418C
game.h includes deck.h and hand.h

```cpp
#include "deck.h"
#include "hand.h"

class game {
  deck d;
  hand h;
  public:
    game();
};
```
game.cpp includes game.h

```cpp
#include "game.h"
#include <iostream>

using namespace std;

game::game() {
    cout << "in game default const" << endl;
}
```
Class Type Member

class Point {
public:
    Point();   //Default Constructor
...
private:
    int x;
    int y;
};
class Points {
public:
    Point p;
};
int main() {
    Points pts;
    cout << pts.p.get_x();
    return 0;
}
class Point {
public:
    Point();  //Default Constructor
...
private:
    int x;
    int y;
};
class Points {
public:
    Points();  //Default Constructor
private:
    Point p;
};
Points::Points() : p() {}
class Point {
public:
    Point();   //Default Constructor
...
private:
    int x;
    int y;
};
class Points {
public:
    Point p[10];
};
int main() {
    Points pts;
    cout << pts.p[0].get_x();
    return 0;
}
Directly calling a constructor on member of another class...

```cpp
#include "game.h"
#include <iostream>

using namespace std;

// constructors are called before {}, this calls non-default for h, instead
// of the default
// game::game() : h(2) {

// this still creates an h that is equivalent to calling the non-default,
// but this calls the default between game() and {}, then calls the
// non-default constructor, then calls the assignment op overload to copy
// into h, and then calls the destructor to for the temp hand after assigning
// to h. This is MUCH less efficient.

game::game() {
  h=hand(2);
  cout << "in game default const" << endl;
}
```

--- INSERT ---
driver.cpp to create a game...

```cpp
#include "game.h"

int main() {
    //game blackjack(); //not an object construction, but function prototype
    game blackjack;    //create an object with default constructor, on ()
    //blackjack=game(); //call a constructor directly after object creation
}
```