Today's announcements

- MP3 is out, **E/C** is due on Tuesday, February 16 @11:59 PM
  MP3 is due on Friday, February 26 @11:59 PM

- **lab_gdb** release is due on Tuesday, February 16 at 11:59 PM

First in-lab exam starts today!

- You **must** come to the lab section to which you are registered
- You **must** bring your i-card with you
- You **must** read Cinda's instructions posted on Piazza or sent through mass-mail
- **No notes, no cell phones, no water bottles, no food. Backpacks with all of the above must be stored in front of the lab.**
- The exam is only 50 minutes.
- You will get at least **1% extra credit** for the exam - as if you attended the lab!
- The purpose is to learn about this system. So we do not want to hurt you by doing this, only extra credit.

- Exam materials: https://chara.cs.illinois.edu/cs225/exams/mt1/

Operator=:

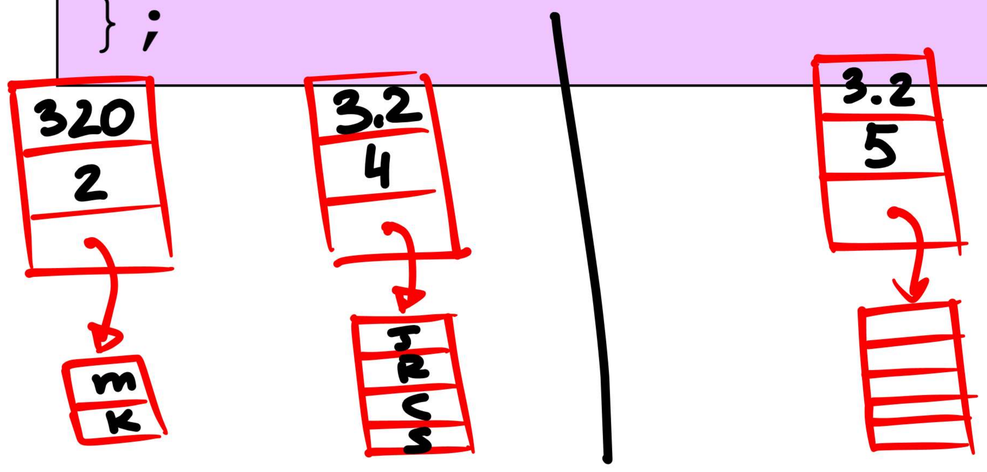
```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & c

~sphere();
sphere & operator=(con

private:

double theRadius;

int numAtts;

string * attributes;

};
```

(2) Why no const? (like in MP2)

```
…

sphere & sphere::operator=(const sphere & rhs){

    if (this != &rhs) {

        clear();
        copy(rhs);

    }
    return *this;

}
…
```

(1) Why not `(*this != rhs)` ?

-
-
-



One more thing..... what is a *const* reference?

```
#include <iostream>
using namespace std;

int main(){
  int x = 5;
  int &y = 7;

  y = 10;
  cout << x << endl;

  return 0;
}
```

```
#include <iostream>
using namespace std;

int main(){
  int x = 5;
  const int &y = 7;

  y = 10;
  cout << x << endl;

  return 0;
}
```

```
#include <iostream>
using namespace std;

int main(){
  int x = 5;
  const int &y = 7;

  x = 10;
  cout << y << endl;

  return 0;
}
```

```cpp
#include <iostream>
using namespace std;

class sphere
{
public:
  //MP2 like operator=
  //const sphere & operator=(const sphere & rhs) {rad = rhs.rad; return *this;}

  //default operator=
  sphere & operator=(const sphere & rhs) {rad = rhs.rad; return *this;}

  double rad;
private:
};
```

1 Should this client code be allowed?

```cpp
int main() {
  sphere a;  a.rad = 20;
  sphere b;  b.rad = 30;

  (b = a);
  cout << b.rad << endl;

  (b = a).rad  = 10;
  cout << b.rad << endl;

  return 0;
}
```

2 How is default operator= implemented?

3 What implementation would protect against it?

Operator=:

```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & c

~sphere();
sphere & operator=(con

private:

double theRadius;

int numAtts;

string * attributes;

};
```

```
…

sphere & sphere::operator=(const sphere & rhs){

   if (this != &rhs) {

      clear();
      copy(rhs);

   }
   return *this;

}
…
```

```
// overloaded operator
sphere & sphere::operator+
    (const sphere & s){




}
```

Usage: c = b + a;

Not: b + a = c;

The Rule of the Big Three:

If you have a reason to implement any one of

- destructor
- copy constructor
- assignment operator

then you must implement all three.

Object Oriented Programming

Three fundamental characteristics:

encapsulation - separating an object's data and implementation from its interface.

inheritance -

polymorphism - a function can behave differently, depending on the type of the calling object.

Inheritance: a simple first example

```
class sphere {
public:
sphere();
sphere(double r);
double getVolume();
void setRadius(double r);
void display();
private:
double theRadius;


};
```

```
class ball:public sphere {
public:
ball();
ball(double r string n);
string getName();
void setName(string n);
void display();
private:
string name

};
```

```
int main() {
sphere a;
cout << a.surfaceArea;}
```

inheritance rules:

-
-
-

```
class sphere {
public:
  void display() {cout << "I am a Sphere!" << endl;}
  double getRadius() {return rad;}
  void setRadius(double r) {rad = r;}
private:
  int rad;
};

class ball: public sphere {
public:
  void display() {cout << "I am a Ball!!!" << endl;}
  string getName() {return name;}
  void setName(string n) {name = n;}
private:
  string name;
};
```

```
void myfun(sphere x){
  x.display();
}
```

```
int main () {
  sphere a;
  ball b;

  a = b;
  b = a;
```

```
int main () {
  sphere a;
  ball b;

  myfun(a);
  myfun(b);

  return 0;
}
```