Announcements

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

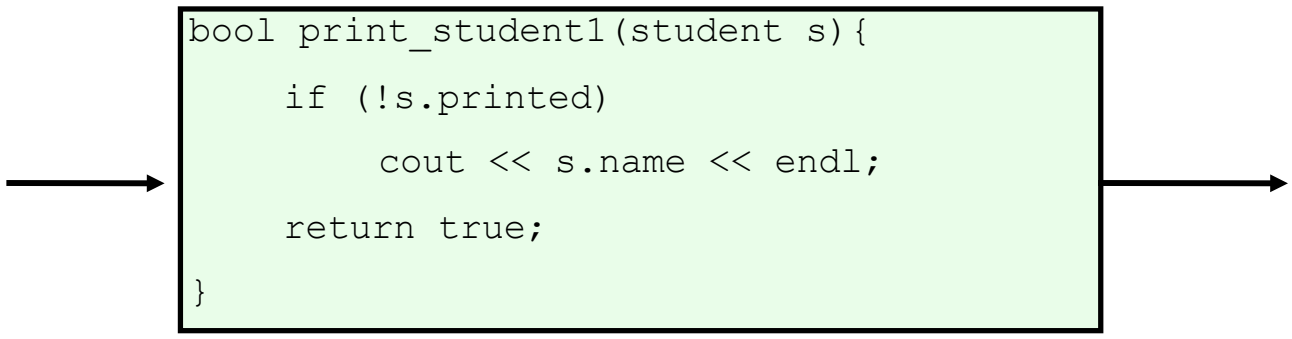
MP2 available, due 2/10, 11:59p. EC due 2/3, 11:59p.

Return values:

What happens when we run code like this:

```
int main() {
    student a;
    print_student1(a);
}
```

?



Return by _____ or _____ or _____ .

Returns:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

Function defn

```
student * print_student5(student s){
    student w = s;
    if (!w.printed){
       cout << w.name << endl;
       w.printed = true;
    }
    return &w;
}
```

Example of use

```
student c;
student * d;
… // initialize c
d = print_student5(c);
```

Returns:

```
struct student {
     string name;
     PNG mug;
     bool printed; // print flag
};
```

Function defn

```
student & print_student5(student s){
     student w = s;
     if (!w.printed){
        cout << w.name << endl;
        w.printed = true;
     }
     return w;
}
```

Example of use

```
student c,d;
… // initialize c
d = print_student5(c);
```

Lesson: don't return 1) a pointer to a local variable, nor 2) a local variable by reference.

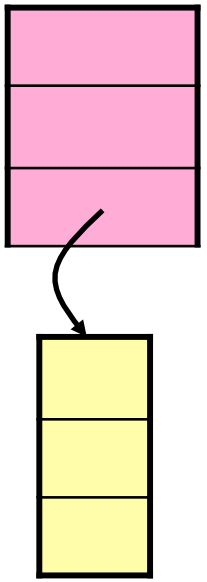
Constructors reprise:

```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & orig);

void setRadius(double newRad);

double getDiameter() const;

…

private:

double theRadius;

int numAtts;

string * atts;

};
```

```
…
//default constructor, alt syntax
sphere::sphere()
{




}
…
```

What do you want the object to look like when you declare it?

```
sphere a;
```



Copy constructor - utility:

```
class sphere{

public:
sphere();
sphere(double r);
sphere(const sphere & orig);
void setRadius(double newRad);
double getDiameter() const;
…
private:
double theRadius;
int numAtts;
string * atts;
};
```

Use 1:

```
sphere myFun(sphere s){
   //play with s
   return s;
}

int main(){
  sphere a, b;
   // initialize a
   b = myFun(a);
   return 0;
}
```

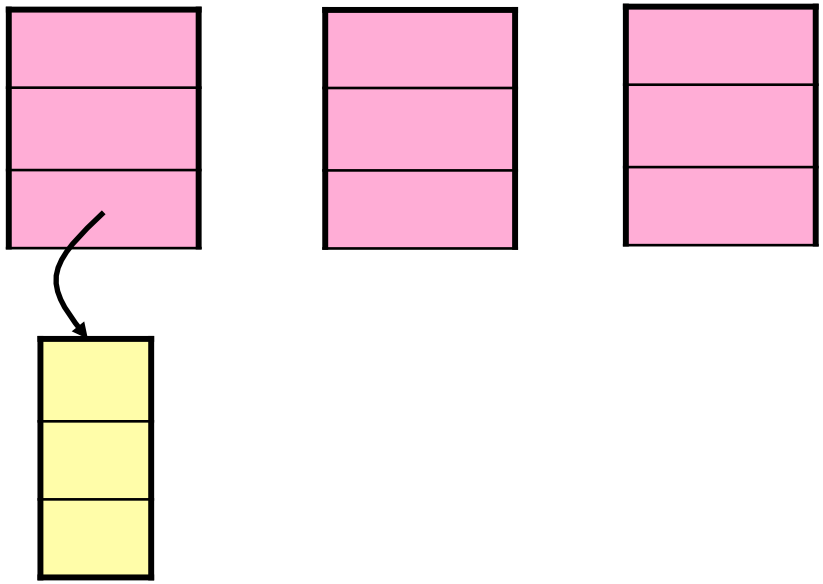
Use 2:

```
int main(){




};
```

Copy constructor:

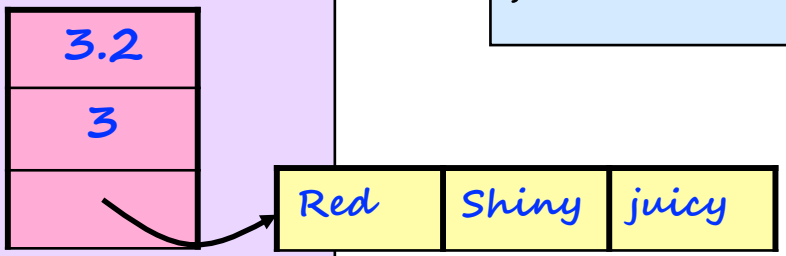
```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & orig);

void setRadius(double newRad);

double getDiameter() const;

…

private:

double theRadius;

int numAtts;

string * atts;

};
```

```
…
//copy constructor
sphere::sphere(const sphere & orig)
{



}
…
```



Destructors:

```
class sphere{
public:
sphere();
sphere(double r);
sphere(const sphere & orig);
~sphere();
…
private:
double theRadius;
int numAtts;
string * atts;
};
```



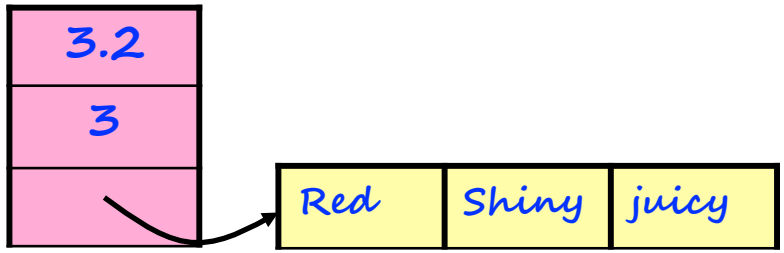

```
void myFun(sphere s){
   sphere t(s);
   …
   // play with s and t
   …
}

int main(){
   sphere a;
   myFun(a);
}
```

```
//destructor
sphere::~sphere(){



}
```

Destructors:

```
class sphere{
public:
sphere();
sphere(double r);
sphere(const sphere & orig);
~sphere();
…
private:
double theRadius;
int numAtts;
string * atts;
};
```

```
int main(){
   sphere * b = new sphere;
   delete b;
   return 0;
}
```




```
//destructor
sphere::~sphere(){



}
```