Announcements

Course policies:

http://cs.illinois.edu/class/cs225

For general assistance:

http://piazza.com/class#spring2013/cs225

MP2 available, due 2/5, 11:59p. EC: 1/29, 11:59p.

Parameter passing so far:

```
struct stu {
     string n;
     PNG mug;
     bool pt; // print flag
};
```

Function defn

```
bool ps1(stu s){
     if (!s.pt)
          cout << s.n;
     return true;
}
```

```
void ps2(stu * s){
     if (!s->pt)
          cout << s->n;
     s->pt = true;
}
```

Example of use

```
stu a;
… // init a
a.pt = ps1(a);
cout << a.pt;
```

```
stu * b;
… // init *b
ps2(b);
cout << b->pt;
```

Parameter passing:

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

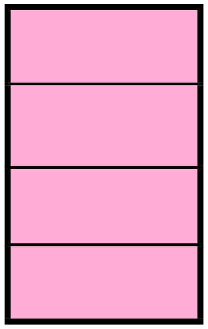
Function defn

```
void print_student3(student    s){
     if (!  s.printed)
          cout <<   s.name << endl;


}
```

Example of use

```
student c;
… // initialize c
print_student3(c);
cout << c.printed << endl;
```



Parameter passing summary:

```
struct stu {
     string n;
     PNG mug;
     bool pt; // print flag
};
```

Function defn

```
bool ps1(stu s){
     if (!s.pt)
          cout << s.n;
     return true;
}
```

```
void ps2(stu * s){
     if (!s->pt)
          cout << s->n;
     s->pt = true;
}
```

```
void ps3(stu & s){
     if (!s.pt)
          cout << s.n;
     s.pt = true;
}
```

Example of use

```
stu a;
… // init a
a.pt = ps1(a);
cout << a.pt;
```

```
stu * b;
… // init *b
ps2(b);
cout << b->pt;
```

```
stu c;
… // init c
ps3(c);
cout << c.pt;
```

Return values:

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

What happens when we run code like this:

```
int main() {
    student a;
    bool b = print_student1(a);
}
```

?

```
bool print_student1(student s){
    if (!s.printed)
        cout << s.name << endl;
    return true;
}
```

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);
```

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

Returns:

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.

Pause for summary:

1. pass/return by value (review)
2. pass/return pointer by value (review)
3. pass/return by reference

Independent learning:

1. flower ** plot;
2. reference variables:

    http://www.cprogramming.com/tutorial/references.html

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;
};
```

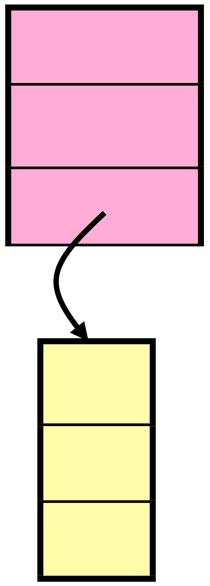
```
…
//default constructor, alt syntax
sphere::sphere()
{



}
…
```

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(){




};
```

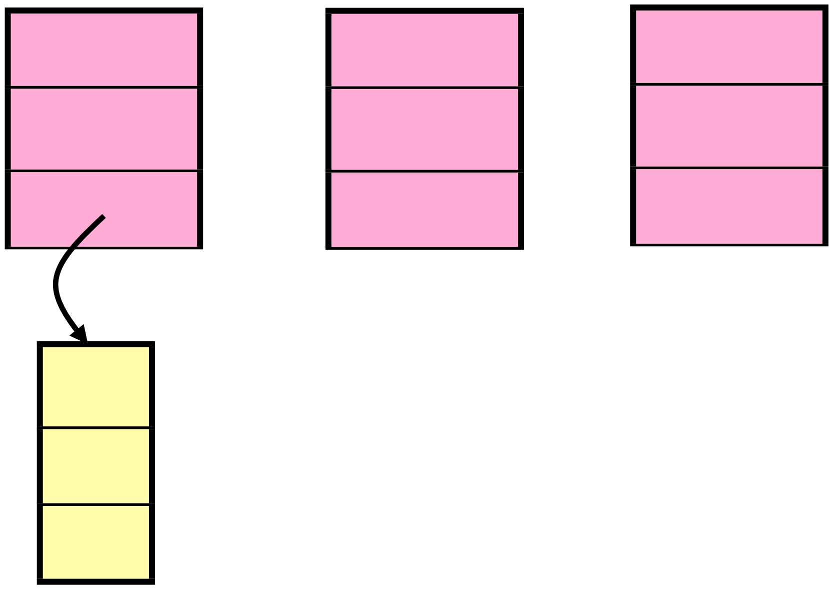
Copy constructor:

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




}
…
```

```
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;
};
```

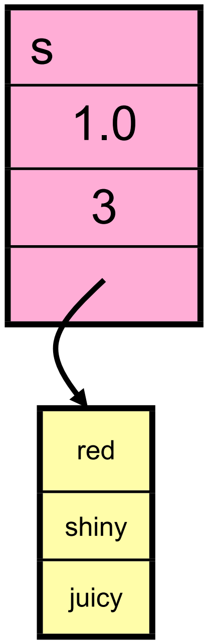


Poser: cctor - why pbr?

```
…
//copy constructor
sphere::sphere(const sphere & orig):
theRadius(orig.theRadius),numatts(orig.numAtts)
{
  atts = new string[numAtts];
  for(int i=0; i<numAtts;i++)
    atts[i]= orig.atts[i];
}
…
```

```
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;
};
```

```
int main(){
  sphere s;
   …// initialize s
   sphere t(s); //invokes CC
   return 0;
}
```


s
1.0
3
red
shiny
juicy