3. Given the Base and Derived classes below, write a constructor for Derived.

```
class Base {
public:
	Base(int x):baseX(x){}
private:
	int baseX;
};

class Derived : public Base {
public:
private:
	int derivedY;
};
```

Code:

```
Derived(int x, int y): Base(x), derivedY(y) {}
```

4. Write a Cookie class which has a constructor, a getDiameter method, overloaded friend output operator (just displays the diameter), and overloaded comparison (<, ==, !=, >) operators. Cookie contains one member variable: diameter (double). Assume everything you need has already been included.

Code:

```
class Cookie
{
	friend ostream& operator<<(ostream& os, const Cookie& p);
public:
	Cookie(double d) : diameter(d) {}
	double getDiameter() const { return diameter; }
private:
	double diameter;
};

ostream& operator << (ostream& os, const Cookie& c){
	os << c.diameter << endl;
	return os;
}

bool operator == (const Cookie& c1, const Cookie& c2){
	return c1.getDiameter() == c2.getDiameter();
}

bool operator != (const Cookie& c1, const Cookie& c2){
	return !(c1 == c2);
}

bool operator < (const Cookie& c1, const Cookie& c2){
	return c1.getDiameter() < c2.getDiameter();
}

bool operator > (const Cookie& c1, const Cookie& c2){
	return !(c1 < c2 || c1 == c2);
}
```