**Name:_____**
**CIS 560 Quiz 2**
**September 2, 2015**

**Question 0: (1 pt)**

```
class Camera;
class Scene {
      Camera cam;
}
Scene* scene1 = new Scene();
delete scene1;
```

In the above example, what is the expected order of destruction of object scene1?

a. ~Scene(); ~Camera(); (Order of destruction is opposite to order of construction)
b. ~Camera(); ~Scene();

**Question 1: (2 pts)**

What is the C++ analogue of writing an *abstract function*? Please provide an example with your answer. What about this function ensures that the programmer will define it in a subclass? Answer both questions in **no more than two sentences**

virtual void Func() = 0;
Must be implemented by subclasses because the compiler will throw an “undefined reference to vtable” error

**Question 2: (2 pts)**
What is the benefit of using initialization lists over assigning member variables values within the body of a constructor? Provide at least **two reasons**.

1. A member variable is initialized with its default constructor even if no initialization list it used, so by assigning it a value in the constructor body you’re essentially initializing it twice.
2. Values passed into the constructors used in initialization lists can be constructed *directly inside* the object named in the list; no local copy is needed.

**Question 3: (2 pts)**

The following code functions as expected, but its design does not take full advantage of C++'s capability to dynamically bind functions. Explain what code you might write *instead* to take advantage of dynamic function binding. Hint: don't use if statements. Your answer should be **no more than three sentences**.

```
class Item{...};
class LightItem : public Item{...};
class HeavyItem : public Item{...};
class Character
{
public:
	void PlayLightPickupAnimation(Item* i);
	void PlayHeavyPickupAnimation(Item* i);

	void PickupItem(Item* i)
	{
		//This statement checks the subclass type of the input Item
		if(LightItem* li = dynamic_cast<LightItem*>(i))
		{
			PlayLightPickupAnimation(li);
		}
		else if(HeavyItem* hi = dynamic_cast<HeavyItem*>(i))
		{
			PlayHeavyPickupAnimation(hi);
		}
	}
};
```

Write a purely virtual function in the Item class that each subclass implements. The body of each implementation should be the body of the IF statement corresponding to the specific subclass, e.g. LightItem::GetPickedUp(Character* c){c.PlayLightPickupAnimation(this);}.

**Question 4: (1 pt)**

In the following code, the developer has made a crucial mistake. Explain the mistake in **no more than two sentences**

```
class Fighter{
public:
      Fighter(){};
      ~Fighter(){};
      virtual void attack();
};

class Swordsman: public Fighter{
public:
      Swordsman(){};
      ~Swordsman(){delete mySword};
      void attack(){..};
private:
      Sword* mySword;
};

int main() {
      Fighter* f = new Swordsman[2];
      delete[] f;

      return 0;
}
```

The destructor of Fighter is not virtual; if s is deleted, then ONLY its Fighter data is properly destructed, not its Swordsman data. Specifically, the mySword variable's memory will not be freed.

**Question 5: (1 pt)**

What will be the output of the following code?

"fighter"

```
class Fighter{
public:
      Fighter(){
            attack();
      };
      virtual void attack() {cout<<"fighter\n";}

      ..
};

class Swordsman: public Fighter{
public:
      ..
      virtual void attack(){cout<<"swordsman\n";};
};

int main() {
      Swordsman s;
      return 0;
}
```