Name: Intho choi **ID:** 1801787

1. Recall the definition for Kvothe from the last Assignment:

```
class Lute {
public:
    Lute(string t) :tone(t){}
    string getTone() { return tone; }
    void setTone(string t){ tone=t;}
private:
    string tone;
};
```

```
class Kvothe {
public:
       Kvothe(int split) :lute("C"),
              num_bindings(split){
              bindings = new int[num_bindings];
       }
       virtual ~Kvothe() { delete[] bindings; }
       virtual void play() { lute.setTone("B#"); }
       void sympathy(int i) { cout << bindings[i]; }
       //...
       Kvothe(const Kvothe &other);//Complete
       Kvothe& operator=(const Kvothe &other);//Complete
private:
       Lute lute;
       int  *bindings;
       int  num_bindings;
};
```

Now consider two more classes, where Kote inherits from Kvothe, and maintains some number of bars.

```
class Bar {
public:
    Bar():drinks(99) { }
    void serve() { drinks--; }
    void restock(int s) { drinks += s; }
    void stock(int d) {drinks = d;}
    int inventory() { return drinks; }
private:
    int drinks;

};
```

```
class Kote : public Kvothe {
public:
   Kote(int nbars):num_bars(nbars) {
       bar = new Bar[num_bars];
   }
   ~Kote() {delete[] bar;}
   virtual void play() {bar[0].serve();}
   void maintain(int b) {bar[b].restock(10);}
   //...
private:
   Bar * bar;
   int num_bars
};
```

a) When we attempt to declare a Kote variable, we get a compiler error. How would you address the issue? The issue is with Kote alone.

The compiler is attempting to call the default constructor of kvothe, but but there isn't one to call. Have Kote explicitly call the parametized constructor for kvothe: e.g. Kote(int nbars): kvothe(6)...

b) Point out the specific ways that Kvothe/Kote exhibit the three properties of inheritance we discussed in class:

Reuse: Kote reuses the function sympathy, and constains member variables, lute, bindings, and num_bindings.

Extension: Kote maintains some number of bars.

Specialization: Both kvothe and kote can play, but they play differently from one another.

Name: Inho Choi **ID:** 1801187

c) Implement the copy constructor for Kote, this is done outside the class definition:

```
Kote::Kote(const Kote& other) : Knothe(other) {
    num_bars = other.num_bars;
    bar = new Bar[num_bars];
    for(int i=0; i<num_bars; ++i)
        bar[i] = other.bar[i];
}
```

d) Overload the assignment operator for Kote, this is done outside the class definition:

```
Kote& Kote::operator=(const Kote& other){
    if(this != &other) {
        Knothe::operator=(other);
        delete[] bar;
        num_bars = other.num_bars;
        bar = new Bar[num_bars];
        for(int i=0; i<num_bars; ++i)
            bar[i] = other.bar[i];
    }
    return *this;
}
```

Name: Info Chot **ID:** 1801787

2. Consider the following program:

```
class A {
public:
	A() :m_msg("Apple") {}
	A(string msg) : m_msg(msg) {}
	virtual ~A() { message(); }
	void message() const {
		cout << m_msg << endl;
	}
private:
	string m_msg;
};
```

```
class B :public A {
public:
	B() :A("Orange") {}
	B(string msg) : A(msg), m_a(msg) {}
	void message() const {
		m_a.message();
	}
private:
	A m_a;
};
```

```
int main() {
	A *b1 = new B;
	B *b2 = new B;
	A *b3 = new B("Apple");
	b1->message();
	b2->message();
	(*b3).message();
	delete b1;
	delete b2;
	delete b3;
}
```

How many times will you see the word Apple in the output? 6

How about Orange? 3

Now assume A's message() is virtual, i.e.,

```
	virtual void message() const …
```

How many times will you see the word Apple in the output? 7

How about Orange? 2