Santa Monica College CS 20A: Data Structures with C++ Assignment 4

Name: Inho Choi **ID:** 1801787

1. Consider the following 7 classes and a main function. What is printed to the console with the complete execution of main?

```
class Shimmy {
public:
	Shimmy(){cout<<"shimmy";}
	~Shimmy(){cout<<"~shimmy";}
};
class Ko {
public:
	Ko() {cout << "koko "; }
	~Ko() { cout << "~koko "; }
private:
	Shimmy sh[2];
};
```

```
class Bop {
public:
	Bop() { cout << "bop"; }
	~Bop() { cout << "~bop "; }
private:
	Ko ko;
};
void main() {
	Bop bop;
	cout << endl;
}
```

shimmyshimmykoko bop
~bop ~koko ~shimmy~shimmy

2. Consider the followingtwo objects, where Kvothe has a Lute which he generally likes to have tuned to "C". You can assume that Kvothe's copy constructor and assignment operator are complete and correct:

```
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):num_bindings(split) {
       bindings = new int[num_bindings];
   }
   ~Kvothe() { delete [] bindings; }
   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;
};
```

a) When we attempt to declare a variable of type Kvothe we get a compiler error. How would you address this? The issue is with Kvothe alone.

The complier is attempting to call Lute's default constructor, but there is no default constructor to call. Have kvothe explicitly call Lute's single parameter constructor.

Name: Inho Choi **ID:** 180n87

b) Implement the copy constructor for Kvothe, this is done outside the class definition:

```
Kvothe::Kvothe(const Kvothe &other) {
    if(other.bindings == nullptr) {
            num_bindings = 0;
             bindings = nullptr;
    }
    else {   num_bindings = other.num_bindings;
                bindings  =  new int[num_bindings];
        for(int i=0; i< num_bindings; i++) {
                bindings[i] = other.bindings[i];
        }
    }
   lute = other.lute;
}
```

c) Overload the assignment operator for Kvothe, this is done outside the class definition:

```
Kvothe&  Kvothe::operator=(const Kvothe &other){
            if(this == &other) return *this;

            delete [] bindings;
      if(other.bindings == nullptr) {
            num_bindings = 0;
             bindings = nullptr;
      }
      else {   num_bindings = other.num_bindings;
                bindings  =  new int[num_bindings];
        for(int i=0; i< num_bindings; i++) {
                bindings[i] = other.bindings[i];
        }
      }
     lute = other.lute;
     return *this;
}
```

d) Implement the destructor for Kvothe, this is done outside the class definition:

```
Kvothe::~Kvothe() {

        delete [] bindings;

}.
```