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

Name: Inho Choi **ID:** 18017887

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;
}
```

✓
Shimmy shimmy koko 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 object creation in kvothe class Lute lute; does not have any parameter and also class Lute does not have default constructor. when we attempt to declare a variable of type kvothe in main, we get a complier error because first call lute's constructor but does not have default constructor.

Name: Inho Choi **ID:** K01780.

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

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

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

```
kvothe&  Kvothe:: operator=(const Kvothe &other){
        if(&other == this){return (*this);}
        delete [] bindings;

       lute =       other.lute;
    num_bindings = other.num_bindings;
       bindings = new int[num-bindings];
         for (int i=0; i<num_bindings; i++){
           bindings[i] = other.bindings[i];
       }

    return (*this);
}
```

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

```
Kvothe::~Kvothe(){
      delete [] bindings;

}
```