CS 32 Solutions Week 2

This worksheet is entirely **optional**, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. Although exams are online this quarter, it is still in your best interest to practice these problems by hand and not rely on a compiler.

Solutions are written in red. The solutions for **programming problems** are not absolute, it is okay if your code looks different; this is just one way to solve the specific problem.

If you have any questions or concerns please contact your LA or go to any of the LA office hours.

Concepts: Copy constructors, assignment operators

Reading Problems

1) What is the output of the following code?

```
class A {
public:
      A()
            { cout << "DC" << endl; }
      A(const A& other)
            { cout << "CC" << endl; }
      A& operator=(const A& other)
            { cout << "AO" << endl; return *this; }
      ~A()
            { cout << "Destructor!" << endl;}
};

int main() {
      A arr[3];
      arr[0] = arr[1];
      A x = arr[0];
      x = arr[1];
      A y(arr[2]);
      cout << "DONE" << endl;
}
```

Time: 5 minutes

Output:

```
DC
```

DC
DC
AO
CC
AO
CC
DONE
Destructor!
Destructor!
Destructor!
Destructor!
Destructor!

2) Find the **4 errors** in the following class definitions so the main function runs correctly.

```
#include <iostream>
#include <string>
using namespace std;

class Account {
public:
  Account(int x) {
    cash = x;
  }
  int cash;
};  // Don't forget the semicolon!

class Billionaire {
public:
  Billionaire(string n) : account(10000){
    offshore = new Account(1000000000);
    name = n;
  }

  ~Billionaire() {
     delete offshore;
  }

  Account account;
  Account* offshore;
  string name;
};

int main() {
  Billionaire jim = Billionaire("Jimmy");
  cout << jim.name << " has "
       << jim.account.cash + jim.offshore->cash << endl;
}
```

```
Output: Jimmy has 1000010000
```

Time: 10 minutes

3) What is the output of the following code:

```
#include <iostream>
using namespace std;

class B {
     int m_val;
public:
     B(int x): m_val(x) { cout << "Wow such " << x << endl; }
     B(const B& other) {
          cout << "There's another me???" << endl;
          m_val = other.m_val;
     }
     ~B() {
          cout << "Twas a good life" << endl;
     }
};

class A {
     int m_count;
     B* m_b;
public:
     A(): m_count(9.5) {
          cout << "Construct me with " << m_count << endl;
          m_b = new B(m_count+10);
     }
     A(const A& other) {
          cout << "Copy me" << endl;
          m_count = other.m_count;
          m_b = (other.m_b != nullptr) ? new B(*other.m_b) :
                                                     nullptr;
     }
     ~A() {
          cout << "Goodbye cruel world" << endl;
          if (m_b)
               delete m_b;
     }
```

```
        int getCount() { return m_count; }
};

int main() {
        A a1, a2;
        A a3 = a2;
        B b1(a3.getCount());
        cout << "Where are we?" << endl;
}
```

Time: 10 minutes

Output:

```
Construct me with 9
Wow such 19
Construct me with 9
Wow such 19
Copy me
There's another me???
Wow such 9
Where are we?
Twas a good life
Goodbye cruel world
Twas a good life
Goodbye cruel world
Twas a good life
Goodbye cruel world
Twas a good life
```

Coding Problems

4) Complete the copy constructor, assignment operator, and destructor of the following class. Be careful to avoid aliasing, memory leaks, and other pointer issues!

```
class A {
        public:
          A(int sz) {
            //...implement this!
          }

          A(const A& other) {
```

```
          //...implement this!
        }

        A& operator=(const A& other) {
          //...implement this!
        }

        //...other functions

        ~A() {
          //...implement this!
        }

      private:
        //one dynamically allocated B object; assume B has a
        //default constructor, a copy constructor, and an
        //assignment operator
        B* b;
        //dynamically allocated array
        int* arr;
        //size of arr (determined by a constructor)
        int n;
        string str;
};
```

Time: 15 minutes

```
class A {
      public:
        A(int sz) {
          b = new B;
          arr = new int[sz];
          n = sz;
        }


        A(const A& other) {
          b = new B(*other.b);
          n = other.n;
          arr = new int[n];
          for (int i = 0; i < n; i++) {
            arr[i] = other.arr[i];
          }
          str = other.str;
        }
        A& operator=(const A& other) {
          if (this != &other) {
            A temp(other);
            std::swap(b, temp.b);
            std::swap(arr, temp.arr);
            std::swap(n, temp.n);
            std::swap(str, temp.str);
          }
          return *this;
        }

   //...other functions

        ~A() {
          delete b;
          delete [] arr;
        }

      private:
        //one dynamically allocated B object; assume B has a
        //default constructor, a copy constructor, and an
        //assignment operator
        B* b;
        //dynamically allocated array
        int* arr;
        //size of arr (determined by a constructor)
        int n;
        string str;
};
```

5) After being defined by the above code, Jim the Billionaire funded a cloning project and volunteered himself as the first human test subject. Sadly, all his money isn't cloned, so his clone has his name, but has $0. Add the needed function to the Billionaire class so the following main function produces the following output.

```
int main() {
  Billionaire jim = Billionaire("Jimmy");
  Billionaire jimClone = jim;
  cout << jimClone.name << " has "
       << jimClone.account.cash + jimClone.offshore->cash
       << endl;
  cout << jim.name << " has "
```

```
        << jim.account.cash + jim.offshore->cash << endl;
}
```

```
Output:    Jimmy has 0
           Jimmy has 1000010000
```

Time: 5 minutes

```
Billionaire(const Billionaire &b)
 : account(0), name(b.name)
{
     offshore = new Account(0);
}
```