COSC 2321 Homework for ch.14

50 pts. total

0] (20 points pencil-and-paper) Assume that we have already developed this class in a library:

```
class Fraction{
    int numer;
    int denom;
public:
     Fraction(){ numer = 0; denom = 1; }
     Fraction(int n, int d){
         numer = n;
         denom = d;
     }
     int getNumer(){ return numer; }
     int getDenom(){ return denom; }
};
```

We now want a new class of **Fraction**-like objects, which, in addition to the functionality above, have the ability to keep track of whether they're simple or not. (A simple fraction has no common factors between its numerator and denominator; simplifying a fraction is dividing both numerator and denominator by the common factor(s) in order to eliminate them.) Let's call the new class **FractionSimpleOrNot**.

Explain briefly how you would use the following tools for the implementation of **FractionSimpleOrNot**:

- Composition:

_____

_____

_____

- Inheritance:

_____

_____

_____

- Multiple inheritance:

_____

_____

Which of the above methods is the most appropriate? Explain:

_____

_____

_____

In the lecture, the following program was used as an example of multiple inheritance[1]:

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

class Person{
    string name;
    int age;
public:
    Person(string n, int a): name(n), age(a){}
    string getName() { return name; }
    int getAge()    { return age; }
};

class Employee{
    bool fullTime;
    string dept;
    float wage;
public:
    Employee(bool f, string d, float w):
                fullTime(f), dept(d), wage(w){}
         bool getFulltime() { return fullTime; }
         string getDept(){ return dept; }
    float getWage()                  { return wage; }
};

class Faculty: public Person, public Employee{
    bool hasTenure;
public:
    Faculty(string name, int age,bool fullTime, string dept, float wage,bool h):
         Person(name, age),
         Employee(fullTime, dept, wage),
         hasTenure(h){}
    void printAll(){
         cout <<"Name:\t\t" <<getName() <<"\tage: " <<getAge();
         cout <<"\t\tFull time: " <<getFulltime() <<endl;
         cout <<"Department:\t" <<getDept() <<"\tWage: " <<getWage();
         cout <<"\tHas tenure: " << hasTenure <<endl;
    }
};

int main(){
         Faculty f("Socrates", 42, true, "Philosophy", 80000, false);
         f.printAll();
}
```

1] ▶ (15 points) Rewrite the class **Faculty** using <u>composition</u> instead of MI.

- Write explicit constructors, including the copy-constructor!
- Test by creating a second object, initializing it, and performing assignment.

2] ▶ (15 points) Rewrite the class **Faculty** using <u>inheritance</u> instead of MI**:** The base (parent) class for **Faculty** should be **Employee** (because it has more code), and the members of **Person** should be added manually.

- Write explicit constructors, including the copy-constructor!
- Test by creating a second object, initializing it, and performing assignment.

<u>Note</u>: all problems require screenshots of source code and output.

[1] The code can be copied from the PDF version of this assignment on our webpage.