Assignment 3

Due Wednesday by 7pm **Points** 120

Constructors should not ask for input from the user - they should just initialize the class data members using the values that were passed as parameters and/or default values. If a data member has a set method, the constructor should use it to initialize that data member, otherwise the constructor can initialize the data member itself.

Remember not to include a main method in the files you submit.

Project 3.a

Write a class called Box that has three double fields called height, width and length. The class should have set methods for each field. It should have a three-parameter constructor that takes three doubles and passes them to the set methods to initialize the fields of the Box. It should have a default constructor that uses the set methods to initialize each field to 1 (using the set methods). It should have a method that calculates and returns the volume of the Box and a method that calculates and returns the surface area of the Box.

The class declaration (interface) and the function definitions (implementation) must be in separate files - the interface or "header" file has a .hpp extension and the implementation has a .cpp extension. As usual, all data members should be private. The Box class might be used as follows:

```
Box box1(2.4, 7.1, 5.0);
Box box2;
double volume1 = box1.getVolume();
double surfaceArea1 = box1.getSurfaceArea();
double volume2 = box2.getVolume();
double surfaceArea2 = box2.getSurfaceArea();
```

Your functions should have the following names:

- setHeight
- setWidth
- setLength
- getVolume
- getSurfaceArea

The files must be named: **Box.hpp** and **Box.cpp**

About using multiple files:

1. Make sure you've read and understood section 7.11.
2. Box.hpp should have "include guards" as discussed on page 447 (use "BOX_HPP").
3. Box.cpp needs to #include Box.hpp. When you include your own .hpp files (header files), put double quotes around them instead of angled brackets. (You should only #include .hpp files, **not** .cpp files.)
4. When testing your program with your own main method, put it in a separate file (this is the "client" code) and give it a name with a .cpp extension.
5. Your main method also needs to #include Box.hpp.
6. If you named the file with your main method "boxMain.cpp", then you can compile your program with "g++ Box.cpp boxMain.cpp -o box".

Project 3.b

Write a class called BankAccount that has a string data member called customerName, a string data member called customerID, and a double data member called customerBalance. The class should have a constructor that takes two strings and a double (name, ID, balance) and uses them to initialize the data members. The data members of this class do not need to be set after they are initialized, so this class doesn't need any set methods - therefore the constructor will directly assign values to the data members instead of calling set methods to do it. The class should have a get method for each data member. It should have a method called withdraw that takes a double parameter and deducts it from the customer balance (the balance is allowed to be negative). It should have a method called deposit that takes a double parameter and adds it to the current balance. The BankAccount class might be used as follows:

```
BankAccount account1("Harry Potter", "K4637", 8032.78);
account1.withdraw(244.0);
account1.withdraw(3012.58);
account1.deposit(37.54);
account1.withdraw(1807.12);
account1.deposit(500.00);
double finalBalance = account1.getCustomerBalance();
```

Your functions should have the following names:

- getCustomerName
- getCustomerID
- getCustomerBalance
- withdraw
- deposit

The files must be named: **BankAccount.hpp** and **BankAccount.cpp**

Project 3.c

Write a class called Point that contains two doubles that represent its x- and y-coordinates. It should have get and set methods for both fields. It should have a constructor that takes two double parameters and passes those values to the set methods to initialize its fields. It should have a default constructor that initializes both coordinates to 0 (using the set methods). It should also contain a method called distanceTo that takes as a parameter a **constant reference** to another Point and returns the distance from the Point that was passed as a parameter to the Point that we called the method of. You will need to use sqrt(). For example at the end of the following, dist should be equal to 5.0:

```
Point p1(-1.5, 0.0);
Point p2(1.5, 4.0);
double dist = p1.distanceTo(p2);
```

Next, write a class called LineSegment that contains two Points that represent its two endpoints. It should have get and set methods for both fields and a constructor that takes two Point parameters and passes them to the set methods to initialize the data members. It should also contain a method called length that returns the length of the LineSegment – by using the distanceTo method on its endpoints – and a method called slope that returns the slope of the LineSegment. You don't need to do anything special for vertical lines - division by zero will result in the special value *inf*, which stands for "infinity". Your program will not be tested with line segments where both endpoints have the same coordinates. The LineSegement class might be used as follows:

```
Point p1(4.3, 7.52);
Point p2(-17.0, 1.5);
LineSegment ls1(p1, p2);
double length = ls1.length();
double slope = ls1.slope();
```

Do not include a main method in the files you submit - just the definition of your Point and LineSegment classes. I will be including a main method for testing, and there can only be one main method in a program. You will of course need to have a main method for testing purposes - just make sure you delete it or comment it out before submitting your files.

The functions for the Point class should have the following names:

- setXCoord, getXCoord
- setYCoord, getYCoord

- distanceTo

The functions for the LineSegment class should have the following names:

- setEnd1, getEnd1
- setEnd2, getEnd2
- length
- slope

The files must be named: **Point.hpp**, **Point.cpp**, **LineSegment.hpp** and **LineSegment.cpp**

Point.cpp and LineSegment.hpp should both #include Point.hpp. LineSegment.cpp should #include LineSegment.hpp. The main method you write for testing will also need to include LineSegment.hpp. If you named the file with your main method "geomMain.cpp", then you can compile your program with "g++ Point.cpp LineSegment.cpp geomMain.cpp -o geom".