Lab 8
Inheritance

Grading:

Package.h	- 5 points if not submitted
TwoDayPackage.h	- 5 points if not submitted
Package.cpp	20 points
TwoDayPackage.cpp	20 points
makefile	5 points
Lab8.cpp (test program)	5 points
Documentation and Style	10 points

Total: 50 points

Submit a zip file containing 6 files: Package.h and Package.cpp, TwoDayPackage.h and TwoDayPackage.cpp, Lab8.cpp, and the makefile.

Material for this lab assignment comes from Chapter 20 in the textbook.

Concepts and Syntax

Inheritance : creating a new class that absorbs (inherits) an existing class's capabilities, then customizes or enhances them.

Base Class (also called parent class or superclass) **:** the existing class

Derived Class (also called the child class or subclass) : the new class

"is - a" relationship: phrase that indicates inheritance

Private

private members of a class are only accessible in the member functions of the class where they are defined.

private members of a base class are inherited, but not directly accessible in a derived class.

Public

public members of a class are accessible in any function.

Class Hierarchy : a diagram showing the inheritance relationships among classes

Types of Inheritance

C++ has three types of inheritance: public, protected, and private.

Public inheritance

To specify that a new class is to be publicly-derived from a base class:

```
class DerivedClassName : public BaseClassName {

   // derived class member definitions

};
```

Public inheritance is the only type of inheritance we will cover this semester.

Calling a Base Class Constructor from a Derived Class Constructor

To specify that a constructor of a derived class is supposed to call the constructor of the base class:

```
// in the constructor definition (not the prototype)

DerivedClassName::DerivedClassName ( parameter list )

: BaseClassName( parameters to be passed to the base class constructor )

{
     // body of derived class constructor function

}
```

Part A:

Create a makefile. Source file names are Package.h, Package.cpp, TwoDayPackage.h, TwoDayPackage.cpp, and Lab8.cpp.

The executable file name should be Lab8.

Programs that do not compile correctly on Linux will receive a grade of zero.
Be sure to test your programs and your makefile on the Linux systems
in the CS department before submitting.

Part B:

Download Package.h and TwoDayPackage.h.

Using the UML Class Diagrams below as your guide,

- Check the prototypes in Package.h to make sure they match the UML. Then create Package.cpp and write the function definitions.
- Check the prototypes in TwoDayPackage.h to make sure they match the UML. Then create TwoDayPackage.cpp and write the function definitions.

Part C:

Write a test program called Lab8.cpp that will instantiate several Package and TwoDayPackage objects. Thoroughly test all of the member functions of these classes.

Package

- senderName : string
- senderAddress : string
- senderCity : string
- senderState : string
- senderZip : string
- recipientName : string
- recipientAddress : string
- recipientCity : string
- recipientState : string
- recipientZip : string
- weight : double
- costPerOunce : double

<< constructor >> + Package (sname : string, saddress : string, scity : string, sstate : string, szip : string, rname : string, raddress : string, rcity : string, rstate : string, rzip : string, w : double, c : double)

+ accessors and mutators for each data member

In the mutator for weight, ensure that weight is a positive value

In the mutator for costPerOunce, ensure that costPerOunce is a positive value.

Mutators should return a reference to the calling object. (Return type for the mutator functions is Package&.)

+ calculateCost( ) : double

The cost to ship a Package is the **weight** (in ounces) times the **costPerOunce** (in dollars).

friend ostream& operator<< (ostream &, const Package&);

TwoDayPackage

(derived from Package)

- flatFee : double

<< constructor >> + TwoDayPackage (sname : string, saddress : string, scity : string, sstate : string, szip : string, rname : string, raddress : string, rcity : string, rstate : string, rzip : string, w : double, c : double , f : double)

This constructor should pass 12 parameters to the Package constructor.

+ setFlatFee (f : double) : TwoDayPackage&

In the mutator for flatFee, ensure that flatFee is positive.

The mutator should return the calling object.

+ getFlatFee ( ) : double

+ double calculateCost( )

The cost to ship a TwoDayPackage is the (**weight** (in ounces) times the **costPerOunce**) plus the flatFee.

friend ostream& operator<< (ostream &, const TwoDayPackage&);