Lab 8
C++ Designing and Implementing Classes

Purpose: Understand the concept of classes as types and objects as instances of a class. Implement class member functions including constructors, accessors, and mutators. Implement a class that illustrates the concept of composition. (A class that has an object as one of its data members.)

Grading:	Documentation & Style (indentation, spacing, etc)	5 points
	Makefile	5 points
	Test Program (tests Date class and Invoice Class)	10 points
	Additional Date Class Functions ▪ Copy Constructor ▪ nextDay ▪ overloaded stream insertion operator <<	6 points
	Invoice Class includes 5 data members and 12 member functions	24 points
		-------------
	Total possible	50 points

You have the completed Date class from Lab 7. Add functions to the Date class

1) a copy constructor

```
Date (const Date &);
```

2) a nextDay function that will increment the day, then adjust the month and year as needed

```
void nextDay( );
```

3) a function that overloads the stream insertion operator <<

This function is not a "member" function of the class. It should be declared as "friend" function of the Date class so that it can access the private data members.

```
friend ostream &operator<<( ostream &, const Date & );
```

The function should display the Date object in the form `mm/dd/yyyy`.

Place the function definition in the Date.cpp file. This function will not have Date:: in front of the function name because it isn't a member function.

See pages 720 – 723 for an example of an overloaded << operator.

Invoice.h

- Design and implement an Invoice class that includes the following member data:

```
Date dateOrdered;
string partNumber;
string description;
int quantity;
float price;
```

Data members are private. All member functions are public.

Follow naming conventions. Use camel case for names of data members and member functions.

There are 12 member functions: a constructor, 5 mutators, 5 accessors, and a toString function.

Remember: For each member function, only the prototype should be placed in the header file. The function definition should be placed in the .cpp file.

- Provide a constructor with five parameters and use the parameter values to initialize the data members (in the order listed above).

  Remember: The best way is to call the mutator functions from within the constructor.

- Provide mutators ("set") functions for all 5 data members. Mutators should validate data as follows:

  quantity cannot be negative
  price cannot be zero or negative
  partNumber and description cannot be the empty string

  If invalid data is passed to a mutator, the mutator should print a message (such as "Quantity cannot be negative.") then prompt the user to enter valid data. The mutator should repeat these steps until valid data is entered.

- Provide accessors ("get") functions for all data members.

- Provide a toString function that returns a nicely-formatted string

```
string toString( );
```

```
Invoice:    mm/dd/yyyy
Part:       partNumber        description
Quantity:   ###               Price:  $ ###.##
```

You may assume that quantity is 3 digits or less. You may assume that price is less than $1000.

Although the + operator can be used to concatenate strings (as in Java), C++ doesn't automatically convert from a numeric type to a string.

Invoice.cpp

- Write the member function definitions – the complete function. Remember to use the binary scope resolution operator with the class name in front of each member function name.

Orders.cpp (This is the test program.)

Create two or more Date objects. Test the new functions: copy constructor, nextDay, and the << operator.

Create several Invoice objects. Include statements to test each one of the Invoice class member functions.

Makefile

Create a makefile that will compile your programs separately, link them, and produce an executable named **orders**.