Programming Project #1 – Classes and Objects

For this project, we are going to create our own classes and use them to build objects. For each class you will create an *.h and *.cpp file that contains the class definition and the member functions. There will also be **one** file called 'main.cpp' that obviously contains the main() function, and will instantiate the objects and test them.

You will create 2 new classes for this project, **one** will be from the list below and **one** will be a required 'Date' class. Here is the list:

WebSite
Shoes
Beer
Book
Song
Movie
TVShow
Computer
Bike
VideoGame
Car

The process is: take your assigned class from the list such as Book. Add to your project new files called Book.h and Book.cpp and create the new class from scratch in those files. I suggest following the simple technique for adding a class discussed in the lecture. Do not overthink. Just come up with the 3 most important data members that describe the class/object you want to build and the write 2 constructors, 3 get member functions, 3 set member functions and a toString method and you are done.

Once you have created a new class, you need to adjust the main() function to allow the user to input the data members for your class and create an object of that type. Use the toString method to then display the new object on the screen. When the project is complete, both classes/objects should be created and displayed with main().

Please do not use the same 3 data types for any class's member variables. That is, do not make them all ' int ' or all ' string '.

As for the Date class, the data members will be int day; int year; string month; Create a Date class with 2 constructors, 3 get member functions, 3 set member functions and a toString method. Adjust the main() function to use the Date class.
When the user wants to build a Date object, on the screen I would input 3 values, one for each member variable. Insert values into the Date object then output a simple message like:

"The Date object is: March 2, 2013"

It is very important that the data input be inserted INTO the object, and pulled back OUT of the object to print the above message.

To ensure you complete each class, use this checklist:

```
_____  Three global variables (not the same type)


_____  Two constructor methods


_____  Three 'get' methods


_____  Three 'set' methods


_____  One 'toString' method


_____  In the main function you create an object, insert values
into the object, and print the object
```

Good Luck

Upgrade the Date Class

There are three (3) levels of understanding of objects. The first is objects are composed of variables and methods. Variables should be private and simple set/get methods allow us to insert data into and pull data back out of the objects.

The second level of understanding is objects should verify their data! To prevent 'Garbage IN, Garbage OUT' the object should NEVER allow bad data to be assigned to the member (or global) variables. Modify the set methods and constructors of the date class to make SURE the day value is always between 1 and 31, make sure the year value is between 1970 and 2099. If the user tries to assign a bad value, print a simple error message and DO NOT change the value of the variable. Example, the 'setDay' is

called with the value 99, a simple message like "Value of day is not in range 1 to 31" should be output and the variable day SHOULD NOT be modified.

The third level of understanding is objects are SMART! Methods can and do generate actions and information that are not explicitly stored in the object. The date class has a string month variable that stores "jan", "feb" ... "dec". Everyone knows month could ALSO be stored as an integer value 1 ("jan") to 12 ("dec"). But we DO NOT need to store BOTH; we can generate the number when needed. Write a method: **int getMonthNumber();** that returns the proper value 1-12 based on the string stored in the month variable. It should return the value -1 if the current value in the month variable is NOT valid. DO NOT store the month number in a member variable (local is OK), generate it when the method is called.

Write a method: **void printDate(int format);** when it is called the format value will determine how the date is output to the screen using cout:

format 0: Mar 12, 2013
format 1: 12 Mar 2013
format 2: 3-12-2013
format 3: 3/12/13

Of course the actual values stored in the object will be output, this is only an example. Modify the main program to print the date in these additional formats. Optional: Use the **int getMonthNumber();** method to check if the user enters a proper value for the month variable.