CS 53 – Introduction to Programming
Homework Assignment #7

Instructions:

(1) This assignment is **due by 9 a.m. on Friday, December 3, 2004**. Late assignments will not be accepted.

(2) You must submit your solution through the *Digital Drop Box* on the Blackboard web site (blackboard.umr.edu). Directions for doing this are given at the end of this document.

(3) This assignment will be worth 5% of your course grade.

This project requires that you write a C++ object class for a *Set*. This particular type of set can only contain integers from 0 to 9 (inclusive). To do this, you are to maintain a "list' of the elements in the set using a one-dimensional array of *bool*. If integer x is a member of the set (x being a value between 0 and 9, inclusive), then elements[x] should be true; otherwise, it should be false. For example, if 5 **is in the set**, then elements[5] should be set to **true**. If 3 **is not in the set**, then elements[3] should be set to **false**.

You are to define your *Set* object class in a file named **Set.h** and define the member functions in a file named **Set.cpp** Posted on the Blackboard web site is a C++ program called **main.cpp** that we will use to test your class. (You are not allowed to make changes to that file!!!) Also posted is a text file of sample output. We will expect to see similar output when we run our main.cpp with your *Set* files.

Specifically, these are the things that you need to define for your *Set* class:

(1) In the *Set* class definition, declare the following two member variables: a *bool* array called ***elements*** of size *maxElements* (where *maxElements* is a global constant that you define as 10), and a *string* called ***name*** for the "name" of the set. Member variables should be declared as *private*.
(2) Write a *Set* ***constructor*** function. It should take as a parameter a string name to be assigned to the set. This function should also assign *false* to each element in the *elements* array. In the constructor function, have it output a message that says that the constructor was called and include the name of the set in the message.
(3) Write a *Set* ***destructor*** function. It should simply output a message that says that the destructor was called and include the name of the set in the message.
(4) Write a member function ***isEmpty***. It should return *true* if the set is empty; otherwise, it should return *false*.

(5) Write a member function ***isElementOf***. Given an integer parameter *x*, it should return *true* if *x* is an element in the set; otherwise, return *false*. Note: If *x* is not between 0 and 9, don't flag it as an error; just return false.
(6) Write a member function ***addElement***. Given an integer parameter *x,* it should make x an element of the set. Note: If x is not an integer from 0 to 9, don't try to add it to the set!
(7) Write a member function ***deleteElement***. Given an integer parameter *x,* it should effectively remove *x* as an element of the set. Note: If *x* is not in the set or is not between 0 and 9, don't flag it as an error; just do nothing.
(8) Write a member function ***set_union***. It should take a *Set s* as a parameter and return a new *Set* which is the union of s and the *Set* it was called for. For example, suppose there is a *Set s1* with elements 1, 2, and 3, and there is a *Set s2* with elements 2, 4, and 5. If we make the call *s3 = s1.set_union(s2)* where *s3* had been declared as a *Set*, then *s3* should have elements 1, 2, 3, 4, and 5.
(9) Write a member function ***set_intersection***. It should take a *Set s* as a parameter and return a new *Set* which is the intersection of s and the *Set* it was called for. For example, suppose there is a *Set s1* with elements 1, 2, and 3, and there is a *Set s2* with elements 2, 3, 4, and 5. If we make the call *s3 = s1.set_intersection(s2)* where *s3* had been declared as a *Set*, then *s3* should have elements 2 and 3.
(10) Write a member function ***print***. It should take a *Set s* as a parameter and print the elements that are in that set. (See the sample output.)
(11) Write an **accessor** member function for *name*.
(12) Write a **mutator** member function for *name*.
(13) Use ***const*** appropriately in your function definitions.
(14) Comment all of your program code!!!

Directions for Submitting Your Assignment for Grading

For this assignment, you are to submit your program using the Digital Drop Box on the Blackboard web site.

To submit your program for grading do the following:

(1) Name your files ***Set.cpp*** and ***Set.h***
(2) Compile and run your program using the GNU (g++) compiler.
(3) Thoroughly test your program on your own.
(4) Login to the CS 53 Blackboard web site (blackboard.umr.edu).
(5) Select *Tools*.
(6) Select *Digital Drop Box*.
(7) Click on *Send File*.
(8) For *Title:* enter **CS 53 – HW #7**
(9) For *File:* use the *Browse* dialog to select one of your files from your computer.

(10) Click on *Submit*.
(11) On the next page that appears (which should say *Receipt: Success*), click on *OK*.
(12) Repeat these steps for the other file you are to turn in.

If you have problems, contact your instructor (leopoldj@umr.edu) or the TA (Rong Zhuge, rzwr6@umr.edu).

You may submit your program any time before the due date, but please try to only submit it once. Do **NOT** wait until the last minute to submit your work in case you encounter problems!