CIS 230 – Spring 2006
Homework #10
Due Friday, April 28, 2006 at 5:00 PM

Submit your files using the submission tool on the qs-server:
/class-files/gilden/230submit

Files to Submit:

- set.h
- set.cpp
- set_test.cpp

Assignment:
Implement a **`Set`** class. Represent a **`Set`** using an array of integers. (In your Set class, include a 32-element array of ints as a private data member.) Assume that the maximum number of Set elements is 32. (That's why we are using a 32-element array.) For the purposes of this assignment, assume that the universal Set, the Set of all possible elements, is the Set of values {0, 1, 2, . . . , 30, 31}. Thus, you should represent the presence of, say, the element 5 in a Set by turning **on** the 5th (counting from 0) element of the array). The Set {3, 12, 20, 5} would be represented by an array with these values:

```
0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
```

For the purposes of this assignment, you need to design functions to implement these Set operations (class methods):

- **Inserting or adding a particular element to the Set**
  To do this you must overload the **istream operator >>** to take a value between 0 and 31 as a parameter and turning **on** the corresponding component of the array representing the Set. (You MUST check to see if the element is already present in the Set – if it is, your method has nothing more to do.) All adding of the member data must be terminated by a sentinal.
- **Deleting a particular element from the Set**
  Taking a value between 0 and 31 as a parameter and turning **off** the corresponing component of the array representing the Set. (You MUST check to see if the element is already absent from the Set – if it is, your method has nothing more to do.)
- **Set union**
  This requires a function which takes two Set objects as parameters and returns a Set which is the union of the two input Sets. Remember that if we have Set A: {3, 6, 8, 10} and Set B: {3, 10, 20}, their union is a third Set: {3, 6, 8, 10, 20}.
- **Set intersection**
  This requires a function which takes two Set objects as parameters and returns Set which is the intersection of the two input Sets. Remember that if we have Set A: {3, 6, 8, 10} and Set B: {3, 10, 20}, their intersection is a third Set: {3, 10}.
- **int operator==(Set &, Set &);**

This **friend** functin returns a 1 if the two sets are equal and a 0 if the sets are not equal. You will need to do an element-by-element comparison of the array elements for each of the two input Sets to determine if they are **exactly** the same or not.

- **Reporting the values of the Set**
  Print the Sets in standard Set notation (ex. Set A: {3, 6, 8, 10}) with the curly brackets and the members separated by commas (do NOT place a comma after the last element in the array). The empty Set would look like this when printed out: { }

You will also need a constructor funciton. Your constructor function should construct an empty Set (that is, a 32-element integer array with all the components valued at "not present" or "off" – remember we are using a value of zero to mean "not present"/"off"). This means that your constructor function will need to assign a zero to every element of the Set object's array.

Write a main() function that:

- Creates 2 Sets, and for each:
  - Allows me to insert elements and uses a sentinel (-1) to stop.
  - Allows me the option to add elements and uses a sentinel (-1) to stop.
  - Allows me the option to delete elements and uses a sentinel (-1) to stop.
- Creates a set that is union of the 2 Sets.
- Creates a set that is an intersection of the 2 Sets.
- Tells me whether or not the 2 Sets are equal.
- Reports the 2 Sets as well as the union and intersection Sets.

Use the file names: set.h, set.cpp, and set_test.cpp

Be sure to have meaningful variable names and instructive user prompts and output labels.

An example of Overloading the istream operator >> with a **friend** function:

```
istream &operator>>(istream &in_data, Rectangle &object1)
{
 for (int index = 0; index < 2; index++)
 {
  cout << "Enter the length for " << index << " ";
  in_data >> object1.length[index];
 }
 return in_data;
}
```