Assignment 3

In this assignment you will implement several different representations of integer sets. A set can be viewed as a container of values, which does not include the same value more than once. Adding a value already contained in the set does not change the set, while removing a value from the set removes the value completely without regard to how many times it was added (if at all).

Download the assignment template "IntSet.cpp" from the course webpage. This file includes an abstract class called IIntSet, which should be implemented by all your classes in this assignment, as well as a main() method that you should not change. The comments in the source file details how each method implementation should work.

3.1 A Simple Integer Set

Create a class called BasicIntSet that represents an integer set using an array of integers. This means that it keeps track of which values are included in the set using this internal container.

3.2 Interval Representation of a an Integer Set

Now write a class that represents integer sets in a more compact fashion, namely by using sets of intervals. For example, the set **{10, 11, 12, 23, 88, 89, 90, 91}** would be represented as the set of intervals { [10..12], [23..23], [88..91]}. Since an interval is more compact than to represent each individual value explicitly, this representation could save a lot of memory in cases with large sets where most values are adjacent.

You should first create a class or struct called Interval that represents a single interval, and then create a class named IntervalIntSet that implements the interface IIntSet and internally retains an array of intervals (or some other container) to keep track of all the intervals.

3.3 Union of two sets

The union of two sets is itself a set that contains all the elements from both sets. For example, the union of the sets {22, 65} and {1, 2, 22, 109} would be the set {1, 2, 22, 65, 109} (note that 22 is present in both the previous sets, but is included only once in the union). The class should internally keep two references to the united sets, and all functionality should be implemented by passing on calls to these two sets. (For example, a value is included in the union above if it is included in either the set {22, 65} or the set {1, 2, 22, 109}.)

3.4 General purpose sets (EXTRA)

The classes that you implemented for this assignment are only useful to create sets of integer values (and not float, double, char, etc). As we have already learned, we could use templates to design the classes in a way that they can be functional with different types of data. Try to rewrite all the classes (and main) in the template version. Try your templates for different types of data (int, float, etc).

NOTE: This is the extra part of the assignment. YOU DO NOT HAVE TO DO IT, but it helps a lot if you do and learn.