Project 1

This project must be accomplished using XP programming style in pairs. That is, you may only work on the project when your partner is present. You and your partner sit at adjacent computers. One person is typing into the code for your project and the other person is monitoring the code, and occasionally, using the other computer to find answers on the Internet. After an hour or so, the partners switch places. Both people must understand all the code all the time. It is academically dishonest to submit code for this project that you neither typed yourself nor supervised being typed.

To have your project graded, you must set up a 45 minute appointment with your partner and the instructor on Friday 10/26, Tuesday 10/30, or Thursday 11/1 to demo your software. At that time, the instructor may ask either partner to explain various design choices. Appointments must be made in class on Wednesday 10/24.

1. Create a class called Interval. The class should include a long start and end value.
   a. In any constructor that takes a start and/or end value, make sure that start < end.
   b. Overload the stream operators >> and << to read and write Intervals. Intervals in a stream should have the format Interval[start, end]
   c. Overload the operators < and <= to mean “start is less” and “start is less or the same”.
   d. Overload the operators > and >= to mean “end is greater” and “end is greater or the same”.
2. Create a class called IntervalSet which contains one or more intervals
   a. Add a union method that takes an Interval and changes the IntervalSet to cover the new Interval. That is
      {Interval[0,2], Interval[5,7], Interval[8,9], Interval[11,15]}.union(Interval[1,8])
      yields {Interval[0,9], Interval[11,15]}
   b. Overload + to combine two IntervalSets to a resulting IntervalSet which covers the same intervals.
   c. Overload – to take out the second set of intervals from the first. That is
      {Interval[0,2], Interval[5,7], Interval[8,9]} - {Interval[0,1], Interval[6,10]}
      yields {Interval[1,2], Interval[5,6]}
3. Write a program that unit tests your Interval and IntervalSet classes.
4. Write a program that uses your Interval and IntervalSet classes and lets you input a people, their departments and weekly availability times (integral hours only). The system then lets you query for times that work for inter-departmental meetings. That is, if you request a meeting with the accounting, printing and mailing departments, the system will give you a (possibly empty) list of times when at least one person from each of those departments can meet. The system should also allow you to change the department for a given person or change that person’s availability (adding times, subtracting times, or replacing the schedule altogether). The system should have some documented menu or set of commands that lets the user intersperse these operations.