**Java**

Briefly familiarize yourself with the ArrayList<> generic class and the List<> and Comparable<> generic interfaces in the Java API (such as found [here](http://docs.oracle.com/javase/7/docs/api/)). To use them, be sure to put

import java.util.\*;

at the top of your Java code.

Put the following in a file named "Part1.java".

1. Define an interface Sorted<> that extends the List<> interface and requires any class implementing the Sorted<> interface to implement a merge method that takes aSorted<> object as a parameter and returns a Sorted<> object. Note that since, as explained below, the merge method is for merging two sorted lists into a sorted list, the elements of a Sorted list must be comparable.
2. Define a generic class SortedList<> that implements both the Sorted and Comparable interfaces, such that two objects of type SortedList<> can be compared using thecompareTo method. You can extend a built-in generic class, e.g. ArrayList, if you want, but that is up to you.

The SortedList class should implement an add method (e.g. by overriding the add() method of ArrayList), so that any new element added to a SortedList is added to the right place in the list in order to keep the SortedList in sorted order (instead of just being added at the end of the list).

The comparison method, compareTo(), required by the Comparable interface, should take another SortedList<T> object (for the same T) and perform a lexicographic comparison, as described in the ML assignment.

You may want to override the toString() method, if you don't like the way your SortedList objects print out.

1. Define a class A that can be used to instantiate SortedList<A>, which also means that two A's must be able to be compared to each other. You can define A any way you like, the only requirements are:
   * A includes a constructor, A(Integer x, Integer y) {...}.
   * When comparing two A objects, the result of the comparison should be based on comparing the sum of the x and y values that the two objects were initially constructed with. That is, given
   * A a1 = new A(7,4);
   * A a2 = new A(5,8);

the result of a1.compareTo(a2) should return -1, indicating that a1 is less than a2 (because 7+4 < 5+8).

You'll also want to override the toString() method, so A objects print nicely.

1. Define a class B that extends A and overrides the inherited compareTo() method. You can define B any way you like, the only requirements are:
   * B includes a constructor, B(Integer x, Integer y, Integer z) {...}.
   * the compareTo method should be overridden so that the value of x+y+z is used as the basis for comparison. For example, given
   * A a1 = new A(6,7);
   * B b1 = new B(2,4,7);
   * B b2 = new B(3,5,8);

the results of the comparisons should be:

a1.compareTo(b1); //returns 0, since (6+7) = (2+4+7)

a1.compareTo(b2); //returns -1, since (6+7) < (3+5+8)

b1.compareTo(a1); //returns 0, since (2+4+7) = (6+7)

b2.compareTo(a1); //returns 1, since (3+5+8) > (6+7)

b1.compareTo(b2); //returns -1, since (2+4+7) < (3+5+8)

You'll also want to override the toString() method, so B objects print nicely.

1. In a separate class named Part1, define the static main() method. In that same class, define a static method, addtoSortedList() that is polymorphic over any type T and which takes two parameters, L and z, where L can be any SortedList into which an object of type T can be inserted and z is of type T. addtoSortedList() should add z toL so that L remains sorted.

1. Finally, in class Part1, put the following method definition.
2. static void test() {
3. SortedList<A> c1 = new SortedList<A>();
4. SortedList<A> c2 = new SortedList<A>();
5. for(int i = 35; i >= 0; i-=5) {
6. addToSortedList(c1, new A(i,i+1));
7. addToSortedList(c2, new B(i+2,i+3,i+4));
8. }
10. System.out.print("c1: ");
11. System.out.println(c1);
13. System.out.print("c2: ");
14. System.out.println(c2);
15. switch (c1.compareTo(c2)) {
16. case -1:
17. System.out.println("c1 < c2");
18. break;
19. case 0:
20. System.out.println("c1 = c2");
21. break;
22. case 1:
23. System.out.println("c1 > c2");
24. break;
25. default:
26. System.out.println("Uh Oh");
27. break;
28. }
29. Sorted<A> res = c1.merge(c2);
30. System.out.print("Result: ");
31. System.out.println(res);
32. }

Have main() call this test() method. The result should look something like:

c1: [[A<0,1> A<5,6> A<10,11> A<15,16> A<20,21> A<25,26> A<30,31> A<35,36> ]]

c2: [[B<2,3,4> B<7,8,9> B<12,13,14> B<17,18,19> B<22,23,24> B<27,28,29> B<32,33,34> B<37,38,39> ]]

c1 < c2

Result: [[A<0,1> B<2,3,4> A<5,6> A<10,11> B<7,8,9> A<15,16> B<12,13,14> A<20,21> A<25,26> B<17,18,19> A<30,31> B<22,23,24> A<35,36> B<27,28,29> B<32,33,34> B<37,38,39> ]]

We'll be testing your code on other test functions, so try different versions of the above test code to see if your code works well.