COS226 Week 3 Activity

1. *Comparables and Static Comparators. Algorithms textbook 2.5*

   (a) Describe a linearithmic algorithm that takes an array of N `Point2D` objects as input, and provides as output a copy of the array with all duplicates removed. The order of the output array does not matter. *Hint:* sort.

   (b) What is the difference between a `Comparable` and a `Comparator`? What methods much each provide?

   (c) Suppose you are sorting Point2D objects by using: `Arrays.sort(pts);` Which comparison method in Point2D is used?

   (d) Write a Java code fragment that sorts using one or more of the static comparators defined in `Point2D (X_ORDER, Y_ORDER, R_ORDER)`

   (e) Show the order of the following points after sorting by Y ORDER using Mergesort. [A(4, 5) B(3, 5) C(4, 2) D(3, 2)]

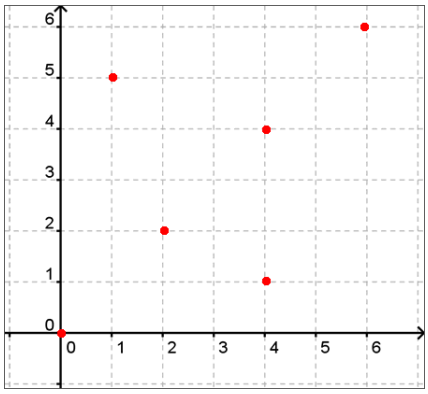
2. *Dynamic Comparators. Algorithms textbook 2.5* For this problem, consider the following code.

```
public class Point implements Comparable<Point>
{
   public static final Comparator<Point2D> X_ORDER = new XOrder();
   public final Comparator<Point2D> SLOPE_ORDER = new SlopeOrder();
   private final double x, y;
   ...

    // compare points according to their x-coordinate
    private static class XOrder implements Comparator<Point> {
        public int compare(Point p, Point q) {
            if (p.x < q.x) return -1;
            if (p.x > q.x) return +1;
            return 0;
        }
    }

   // compare points according to their distance to this point
   private class SlopeOrder implements Comparator<Point> {
       public int compare(Point2D p, Point2D q) {
              //returns -1 if slope from this to p is less than this to q
              //returns +1 if greater
              //returns 0 otherwise
              //the slope from a point to itself is negative infinity
       }
   }
}
```

(a) What is the difference between a static and a dynamic `Comparator`?

Suppose we have an array of points created by the declaration below.

```
Point[] pts =  new Point[6];
pts[0] = new Point(0, 0);      pts[1] = new Point(1, 5);
pts[2] = new Point(2, 2);      pts[3] = new Point(4, 1);
pts[4] = new Point(4, 4);      pts[5] = new Point(6, 6);
```



(b) Label the points with the order that results from sorting the points by their slope to point 0 (similar to how we labeled the points during the Graham Scan part of lecture). For example, if point (1, 5) ends up in the 1st position, it should be labeled with a 1.

(c) Write code that sorts these points by their slope to pts[0]. This accomplishes the sort from part b.

(d) After sorting by (0,0), the array should be in the order shown in bracket notation below.

[(0,0), (4, 1), (2, 2), (4, 4), (6, 6), (1, 5)]

Give the order of the array in bracket notation if we next sort with respect to (1,5)'s slope order. Start from the ordering in the problem above.

[ ]

(e) Now give the results if we stably sort your answer to part d with respect to (2, 2)'s slope order.

[ ]