10 Arrrrghrays (25 pts)

Purplebeard and his lackey Turquoisenail are sailing the 10 seas. In order to sail well, they want to be able to create a map. They managed to create their **square** map, but Turquoisenail tripped and put it through a paper shredder. They managed to store the scrap images into a 1d array, but they need to piece it back into an NxN map. You are lucky because on each piece you have the longitude and latitude written down. Write a short program to help put the pieces back together. The pieces should be as follows:

same lat →

0,20	10,20	20,20
0,10	10,10	20,10
0,0	10,0	20,0

(long, lat)

In the upper left corner of the table, 0 is the longitude and 20 is the latitude.

For this problem you have access to only arrays

a For the first part of this problem, make a Piece class that store longitude and latitude.

```
public class Piece{
    public int longitude;
    public int latitude;

    ------------------------------------
    public Piece(int x, int y){
        longitude = x;
        latitude = y;

        ------------------------------------
    }
}
```

b The next part of this problem is take the Pieces in the given 1D Piece array, where Pieces are in no particular order, and put it into a 2D array where each row filled with Pieces that have the same latitude.

```
public Piece[][] groupByLat(Piece[] p){
    int width = (int) Math.pow(p.length, 0.5);
    Piece[][] latGroup = new Piece[width][width];
    for(int i = 0; i < width; i++){
        for(int j = 0; j < width; j++){
            if(latGroup[j][i]==i){
                ;
                break;
            }
            else if(){
                int counter;
                for(counter=0; counter < p.length-1; counter++){
                    if( j == p[counter].latitude && i==p[counter].longitude ){
                        break;
                    }
                }
                latGroup[j][i] = p[counter];
                break;
            }
        }
    }
return latGroup;
}
```

c Our goal is to now to complete the process of taking in a 1D unsorted Piece array and transform it such that it becomes a sorted 2D array as shown on the first page of this problem (longitudes increase from left to right and latitudes increase from down to up). To complete this problem you have the following methods.

- *groupByLat(Piece[] p)*: From part b, takes in a 1D Piece array and converts it into a 2D Piece array where Pieces share a row if they have the same latitude.
- *sortbyLat(Piece[][] p)*: Takes in a 2d array of Pieces and returns it sorted in the correct order such that the row that contains the smallest latitudes is at the 0th index.
- *sortHalfLong(Piece[] p)*: Takes in a 1D array of Pieces and **half sorts** them all by longitude. In this problem, the term half sort means that the array is fully sorted except the first half of the sorted array is switched with the second half of the sorted array. For example:
  say we have an array $[9, 2, 4, 0]$. This array sorted would be $[0, 2, 4, 9]$. This array half sorted would be $[4, 9, 0, 2]$ since the first half of the sorted array, $[0, 2]$, would be swapped with the second half, $[4, 9]$.

```
public Piece[][] solvePuzzle(Piece[] scattered){

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------

    ------------------------------------------------
}
```