CS6140: Advanced Programming Lab
Assignment 5 : Finding the Closest Pair of Points
An Exercise in Divide and Conquer Strategy

Lead TA: Dhannya

September 18, 2014

Input

You will be given a set of 2D points with integer coordinates. More specifically, your program should take the name of a file in command line and that file must contain the points set in the following manner. Each line contains the details of a single point, which are (i) its x-coordinate and (ii) its y-coordinate. After all the points (one per line), the final line must only contain the string "done". (You may assume that the total number of 2D points given as input will be ≤ 200). For example:

```
3 56
91 40
90 34
78 20
98 20
76 5
done
```

1 Output

If the input is not conforming to the input format, then you must print an appropriate error message. Exit if there are errors. Otherwise, you must print a single integer to the console, which must be the Manhattan distance[1] between a pair of points closest to each other in Manhattan distance. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is simply $|x_1 - x_2| + |y_1 - y_2|$. Intuitively, it is the distance from one point to the other if you are restricted to go up/down or left/right, but never at any other angle. Think of a taxi

[1]This is also called the taxicab distance, or rectilinear distance, or, for the more mathematically inclined, the ℓ_1 distance. See `http://mathworld.wolfram.com/TaxicabMetric.html` for an illustrative figure.

driver trying to go from one point to another in Manhattan, a borough in New York with an almost perfect grid-like road network.

2 The Divide and Conquer Requirement

The following is a very simple algorithm to solve this problem. It will fetch you no marks.

```
min = a very large number
for i = 1 to n-1
   for j = i+1 to n
    d = Manhattan distance between ith and jth points
if d< min, then min = d
Print d.
```

You must solve this problem using divide and conquer and the algorithm employed must be an $O(n \log^2 n)$ time algorithm. The technique for a slightly different problem (using Euclidean distance instead of Manhattan distance) is described at `http://www.cs.mcgill.ca/~cs251/ClosestPair/index.html` (sections 1, 2, and most importantly 3). First, read those three sections and understand how the algorithm works for Euclidean distance. Then, think carefully about how the algorithm can be adapted to Manhattan distance.

3 Log File

You should maintain a comparison log file "comparisons.log". The filename is fixed and in each execution, create the file afresh, i.e, wipe out any content that might already be in the file. Each time you calculate the Manhattan distance between two numbers, add a line in the log file with the coordinates of the two points — nothing more. So for example, if you computed the Manhattan distance between $(5, 78)$ and $(9, 56)$, your entry should read:

```
5 78 9 56
```

4 What to upload

For simplicity, put all your classes and functions in one cpp file called `<rollnumber>.cpp` and upload it into moodle.