Closest Pairs

(3 points.) To start, run `python new-homework.py [language] closest-pair`

Task. Given a set of points in $\mathbb{R}^2$, $\mathcal{P} = \{p_1, \ldots, p_n\}$ with $p_i = (x_i, y_i)$, we want to find the closest pair of points, that is, the $p_i \neq p_j$ that minimizes the Euclidean distance $\|p_i - p_j\|$. The goal of this exercise is to devise a divide-and-conquer algorithm for this problem.

Consider the following sketch:

1. Find a value of x that splits $\mathcal{P}$ in half according to $x_i < x$ or $x_i \geq x$. Call these split sets $\mathcal{L}$ and $\mathcal{R}$.
2. Recursively find the closest pair of points in $\mathcal{L}$ and $\mathcal{R}$. Call these pairs $p_L, q_L \in \mathcal{L}$ with distance d_L and $p_R, q_R \in \mathcal{R}$ with distance d_R, respectively.
3. Determine if there is a point in $\mathcal{L}$ and a point in $\mathcal{R}$ that are closer than $d = \min(d_L, d_R)$. (Is there a way to limit the set of points that must be examined? See hints below.)
4. Return the closest pair found and (since you have it) the distance between them.

Fill in the gaps in this sketch and implement this algorithm.

You should write a function *`closestPair`* that takes a collection of two-dimensional points (list, set, matrix, or any representation that you find easiest) and returns the closest pair of points and their distance. A data file `closest-pairs1.txt` is installed in the `closest-pair/Data/` directory when you start the assignment.

Questions. Briefly answer the following questions in comments on your GitHub pull request.

1. How does your function behave if there are several pairs of points with the same distance? (This could happen easily, say, with integer-valued data.)
2. How do you think it *should* behave?
3. And how would you alter your code to produce this desired behavior?

Requirements.

☐ Write the function *`closestPair`* matching the above specification.

☐ Supply a test suite for your functions.

☐ Call your function with the data from `closest-pairs1.txt` and report the results in your pull request. You can do this interactively or with a script. The closest pair of points should be roughly 1.247×10^{-5} apart.

Hint 1. What points can you discard in step 3?

Hint 2. Convince yourself that any $d \times d$ square in the plane can contain at most four points.

Hint 3. Use S to show that in Step 3, if you sort the remaining points by their y-coordinates, you need only compute the distance to each point on the list to the seven subsequent points.