605.621 Foundations of Algorithms Fall 2020
Programming Assignment 1
Assigned with Module 1, Due at the end of Module 6 (October 12)

The goals of Programming Assignment 1 are: (1) to have you prepare your programming tools to support algorithm implementation to measure asymptotic behavior of your algorithm implementation, and (2) to exercise the algorithm analysis techniques you'll study in your Modules. A firm foundation in these two goals will set you off towards successful assignment work this semester.

This programming problem is not a collaborative assignment. You are required to follow the **Programming Assignment Guidelines and Pseudocode Restrictions** (Blackboard page *Syllabus & Course Information*) when preparing your solutions to these problems.

This assignment requires you to "construct an algorithm". There are, of course, many algorithms solving this problem. Some algorithms are asymptotically more efficient than others. Your objectives are to (a) complete the assignment, and (b) submit an asymptotically efficient algorithm.

1. **Conditions**
   (a) Consider a set, P, of n points, $(x_1, y_1), ..., (x_n, y_n)$, in a two dimensional plane.
   (b) A metric for the distance between two points (x_i, y_i) and (x_j, y_j) in this plane is the Euclidean distance $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.
2. **Closest Pairs** [90 points]
   (a) [40 points] Construct an algorithm for finding the $m \leq \binom{n}{2}$ closest pairs of points in P. Your algorithm inputs are P and m. Return the distances between the m closest pairs of points, including their x and y coordinates.
      i. [25 points] Define your algorithm using pseudocode.
      ii. [15 points] Determine the worst-case running time (page 25) of your algorithm (call this the algorithm's worst-case running time).
   (b) [20 points] Implement your algorithm. Your code must have a reasonable, consistent, style and documentation. It must have appropriate data structures, modularity, and error checking.
   (c) [10 points] Perform and submit trace runs demonstrating the proper functioning of your code.
   (d) [10 points] Perform tests to measure the asymptotic behavior of your program (call this the code's worst-case running time).
   (e) [10 points] Analysis comparing your algorithm's worst-case running time to your code's worst-cast running time.
3. **Retrospection** [10 points]
   (a) [10 points] Now that you have designed, implemented, and tested your algorithm, what aspects of your algorithm and/or code could change and reduce the worst-case running time of your algorithm? Be specific in your response to this question.