**Project 3 Part 4 Randomized algorithm (average O(N))**

[Project 3 Part 4 Randomized algorithm (average O(N))](https://fcps.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_1516674_1&content_id=_43601203_1&mode=reset)

**Part 4: create a method part4() that**

The next lab is to implement the following algorithm to find the closest pair of points:

[ClosestPairRandomized.pdf](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-41494897_2/xid-41494897_2) [ClosestPairRandomized.pdf - Alternative Formats](https://fcps.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_1516674_1&content_id=_43601203_1&mode=reset)

Also a visual simulation Created by Dr. Torbert:

[Closest Pair.mp4](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-48827043_2/xid-48827043_2)

Before you start you should read the points from the file points.txt into 2 vectors(this part should not be timed). One vector will be used by part3 and the other vector will be used by part4 (you may have as one parameter to this methods a reference to the vector.

In part 4:

a) shuffle the vector using the knuth shffle we discussed in class

b) Complete the randomized algorithm described in the document above

c) Make sure your timing includes the shuffle and the randomized algorithm

In the main you should:  
a) call part3(...)  
b) call part4(..)  
c) display on the screen and in the results.txt the 2 points and minimum distance obtained for both approaches also the time to complete each approach (you may do this either by creating global variables or by making both part3 and part4 to return some result)

Please turn in to Mr. Jurj a printout of the following file filled:

[Project 3 Part 4.docx](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-49038442_2/xid-49038442_2) [Project 3 Part 4.docx - Alternative Formats](https://fcps.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_1516674_1&content_id=_43601203_1&mode=reset)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A) You may use the following files to test:

[points1k.txt](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-41750517_2/xid-41750517_2)

[points10k.txt](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-41750518_2/xid-41750518_2)

[points100k.dat](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-41750519_2/xid-41750519_2)

[points1m.txt](https://fcps.blackboard.com/bbcswebdav/pid-44364004-dt-content-rid-41750522_2/xid-41750522_2)

B) use the following link if you need help to make an unordered map of pairs:

   https://www.geeksforgeeks.org/how-to-create-an-unordered\_map-of-user-defined-class-in-cpp/

<https://www.geeksforgeeks.org/how-to-create-an-unordered_map-of-pairs-in-c/>

C) for the pair (x,y) of the grid use "unsigned long long" so you don't run over the limit of int