**Project 3 Part 2 (intermediary recursive approach)**

Name: Michael Fatemi Period: 7

Date: 11/6/2020

Is your lab name l032?(lowercase L followed by digits 032) Yes

Did you created a class to store a point? Yes

Did you use a vector to store the points you generated? Yes

Did you use at least one iterator to traverse the vector you created? Yes

Does your main contain only 2 calls of: part1() and part2()? Yes

Paste here a clear picture of the graph that compares the running times of the two algorithms versus number of points. (use 2 different colors for the 2 graphs):

|  |  |  |
| --- | --- | --- |
| **Points** | **Brute force** | **Divide and conquer** |
| 100 | 1 | 0 |
| 200 | 3 | 1 |
| 400 | 13 | 1 |
| 800 | 52 | 2 |
| 1600 | 206 | 4 |
| 3200 | 829 | 9 |
| 6400 | 3580 | 20 |
| 12800 | 14483 | 49 |

As we can see, the brute force method scales quadratically with the number of points, as we would expect. Conversely, the divide and conquer method scales just barely more than linearly, which we would expect from an O(n log n) algorithm. It is able to process over 12,000 points in under the time it takes the brute force method to process 800 points. The second method is more stable and should be used in situations where performance is critical.

***Graphs are on the next page***



