**CSC 3430 Algorithm Design and Analysis**
**Homework 5: Implement Closest Point Brute-Force and Divide&Conquer Algorithms**
**Due: Friday, March 3, 2017**

Design and implement a C/C++ program that compares the performance of the brute-force algorithm and the DivideCconquer algorithm for determining the Closest Two Points in a 2D plane for an array of unique points [x,y] with integer coordinates within the plane.

For each algorithm, the output must specify

1. The shortest distance between the two closest points.
2. The index within the original Point array and the [x,y] coordinates for each of the two points.
3. The number of distance calculations between 2 points that the algorithm required.

Some details and requirements:

- An Windows executable "ClosestPoints.exe" is on Canvas that outputs slightly more detail as it runs.
- It is recommended that you study and refer to the following online article as you begin work on this lab. You may use any of the ideas or code detail in these articles that you like, although this is not a requirement for this lab assignment. If you do use specific code details, be sure to include appropriate acknowledgements in your code.
  http://www.geeksforgeeks.org/closest-pair-of-points/
  A few observations about the code in this example:
  - The dist( ) function calls sqrt( ) for it's computation, which assumes a float argument. Since this lab uses integer coordinates for the points, it is necessary to manually force the conversion. Suggested: change p1.x to float(p1.x).
  - The closestUtil( ) function declares a local variable "Point strip[n];", which is not standard in older versions of C/C++. Instead you can change to "Point *strip = new Point[n];", and then be sure to "delete [ ] strip;" before returning from the function.
- You'll have to use a global variable and related code to count the number of distance calculations (e.g., the dist() function) required for an algorithm. Initialize to 0 before performing an algorithm and have dist( ) count each time it is executed, so that at the end you can output the number of comparisons required for that algorithm.
- You must modify the base code to keep track of which 2 points were the closest.
- To initialize a large array with random unique points, use something like;

```
#include <time.h>
void initpoints(Point P[], int n, int range, int seed=-1)
{
    if (seed < 0) seed = clock();
    srand(seed);

    // Fill array with unique points[i] for i from 0 .. n-1
    int i = 0, j;
    while (i < n)
    {
        Point p(rand() % range, rand() % range);  // Create random point within range

        for (j = 0; j < i; ++j) {  // Search to see if new point is unique
            if (P[j].x == p.x && P[j].y == p.y) break; // already existing point
        }
        if (j >= i) {
            P[i] = p;  // Unique point -- save it
            i++;               // Move on to next point
        }
    }
}

Point Pbig[100];
initpoints(Pbig, 100, 1000, 0);
```

**Turn in on due date:**

1) Source code listings for all of your solution code.
2) Execution runs output for the 4 small datasets shown below, plus one size-100 random dataset and one size-1000 random dataset. This can be done as 6 separate runs, or a larger main() that contains multiple datasets.

```
--------------Closest Points in a plane analysis: Mike Tindall--------------
Points[6] -- Mike Tindall Analysis
[   2,   3], [  12,  30], [  40,  50], [   5,   1], [  12,  10], [   3,   4]
Brute Force
The smallest distance is 1.414214 --> P0[2,3] : P5[3,4]
 -->     15 point-to-point distance calculations
Divide & Conquer
The smallest distance is 1.414214 --> P0[2,3] : P5[3,4]
 -->      6 point-to-point distance calculations
-----------------------------------------------------------------MikeTindall
Points[6] -- Mike Tindall Analysis
[   1,   1], [   2,   5], [  15,   1], [  16,   1], [  36,   4], [  39,   1]
Brute Force
The smallest distance is 1.000000 --> P2[15,1] : P3[16,1]
 -->     15 point-to-point distance calculations
Divide & Conquer
The smallest distance is 1.000000 --> P3[16,1] : P2[15,1]
 -->      7 point-to-point distance calculations
-----------------------------------------------------------------MikeTindall
Points[6] -- Mike Tindall Analysis
[   1,   1], [   2,   5], [  15,   1], [  23,   1], [  36,   4], [  39,   1]
Brute Force
The smallest distance is 4.123106 --> P0[1,1] : P1[2,5]
 -->     15 point-to-point distance calculations
Divide & Conquer
The smallest distance is 4.123106 --> P0[1,1] : P1[2,5]
 -->      6 point-to-point distance calculations
-----------------------------------------------------------------MikeTindall
Points[6] -- Mike Tindall Analysis
[   1,   1], [   2,  24], [  15,   1], [  23,   1], [  36,   4], [  39,   1]
Brute Force
The smallest distance is 4.242640 --> P4[36,4] : P5[39,1]
 -->     15 point-to-point distance calculations
Divide & Conquer
The smallest distance is 4.242640 --> P4[36,4] : P5[39,1]
 -->      6 point-to-point distance calculations
-----------------------------------------------------------------MikeTindall
Points[100] -- Mike Tindall Analysis
[ 719,  38], [ 437, 238], [ 797, 855], [ 285, 365], [ 612, 450], [ 100, 853]
[ 281, 142], [ 921, 537], [ 285, 945], [ 680, 997], [ 891, 976], [ 906, 655]
.............
[ 452, 910], [ 156, 248], [ 475, 911], [ 719, 436], [ 255, 388], [ 840, 447]
[ 493, 854], [ 841, 755], [ 167, 204], [ 556, 816], [ 927,   1], [ 551, 214]
[ 279,  83], [ 846, 674], [ 799,  14], [ 537, 665]

Brute Force
The smallest distance is 10.000000 --> P40[152,673] : P65[160,667]
 -->   4950 point-to-point distance calculations

Divide & Conquer
The smallest distance is 10.000000 --> P65[160,667] : P40[152,673]
 -->    128 point-to-point distance calculations
-----------------------------------------------------------------MikeTindall
Points[1000] -- Mike Tindall Analysis
[7719,  38], [2437,1238], [1797,8855], [2285,8365], [ 612, 450], [8100,5853]
[ 281,1142], [5921, 537], [6285,8945], [4680,2997], [1891, 976], [5906,1655]
.....................................
.....................................
.....................................
.....................................
...
[9446,2947], [6983, 326], [4905,7606], [9014,1642], [8143,7304], [5151,4275]
[ 426,9908], [ 719,1575], [1204,7200], [ 785,7357], [8542, 375], [3448,5472]
[2759, 890], [4840,6658], [5712,8450], [2773,  51]

Brute Force
The smallest distance is 1.414214 --> P445[5020,9141] : P686[5019,9140]
 --> 499500 point-to-point distance calculations

Divide & Conquer
The smallest distance is 1.414214 --> P686[5019,9140] : P445[5020,9141]
 -->   1091 point-to-point distance calculations
-----------------------------------------------------------------MikeTindall
```