BLG 336E - Analysis of Algorithms II
2020/2021 Spring
Final Project
Question 2

- Please submit your homework only through Ninova. Late submissions will not be accepted.
- Please do not forget to write your name and student ID on the top of each document you upload.
- You should write your code in C++ language and try to follow an object-oriented methodology with well-chosen variables, methods, and class names and comments where necessary.
- Your code should compile on Linux using g++. You can test your code through ITU's Linux Server.
- Because your codes will be processed with an automated tool, `Calico`, make sure that your output format matches the given sample output.
- You may discuss the problem addressed in the homework at an abstract level with your classmates, but you should not share or copy code from your classmates or any web sources. You should submit your individual homework. In case of detection of an inappropriate exchange of information, disciplinary actions will be taken.
- If you have any questions, please ask on related message board named **Final Exam Q2 Thread** for this question.

1 Finding Distance Between the Closest Pair of Points using Divide and Conquer Technique (25 pts)

Implementation [25 pts]
In this question, you are expected to implement the "Closest-Pair" algorithm given in Figure 1. "The closest pair of points" problem basically finds the closest pair among a list of points in the given space [1]. Please check the textbook [1] and course slides for

further details on this topic.

```
Closest-Pair(P)
  Construct P_x and P_y  (O(n log n) time)
  (p*_0, p*_1) = Closest-Pair-Rec(P_x,P_y)

Closest-Pair-Rec(P_x, P_y)
  If |P| ≤ 3 then
    find closest pair by measuring all pairwise distances
  Endif

  Construct Q_x, Q_y, R_x, R_y (O(n) time)
  (q*_0,q*_1) = Closest-Pair-Rec(Q_x, Q_y)
  (r*_0,r*_1) = Closest-Pair-Rec(R_x, R_y)

  δ = min(d(q*_0,q*_1), d(r*_0,r*_1))
  x* = maximum x-coordinate of a point in set Q
  L = {(x,y) : x = x*}
  S = points in P within distance δ of L.

  Construct S_y (O(n) time)
  For each point s ∈ S_y, compute distance from s
     to each of next 15 points in S_y
     Let s, s' be pair achieving minimum of these distances
     (O(n) time)

  If d(s,s') < δ then
     Return (s,s')
  Else if d(q*_0,q*_1) < d(r*_0,r*_1) then
     Return (q*_0,q*_1)
  Else
     Return (r*_0,r*_1)
  Endif
```

Figure 1: Pseudocode for the Closes-Pair algorithm [1]

In this assignment, you will work in two-dimensional space. In other words, you will deal with a list of points that each one has two coordinates (x and y).

Your implementation should take a list of points as input and produce the distance between the closest pair of points as output. A skeleton code "**q2.cpp**" is provided with the question. The skeleton code already includes class definitions and some steps for distance calculations, file reading, and output formatting. But the code file does not include the main calculation for the given problem. You need to implement the main part of the algorithm by filling empty spaces in the code file specified with the "FILL HERE' comment. You may define new functions if necessary.

Three test files **points7.txt**, **points10.txt**, and **points20.txt** are provided with the question. Each test file includes a certain number of points: 7, 10, and 20 points, respectively. The first row in each file corresponds to the number of points in the space, while the rest of the rows correspond to the list of points. Each point is separated by a new line character. Also, each point is represented by two coordinates that are separated by a space character.

Sample commands for the compilation and test are as follows:

```
g++ -std=c++11 q2.cpp -o q2
./q2 points7.txt
./q2 points10.txt
./q2 points20.txt
```

The expected output of your code includes the distance between the closest pair of points. Figure 2, 3, and 4 show the expected outputs for the given test cases.

```
[beyzaeken@ssh Beyza Eken Q2]$ g++ -std=c++11 q2.cpp -o q2
[beyzaeken@ssh Beyza Eken Q2]$ ./q2 points7.txt
Points coordinates (x y):
1       10
22      20
20      52
1       1
32      5
5       7
8       65
Total number of points: 7
Distance between the closest points: 5
```

Figure 2: Compilation and the output for test file "points7.txt".

Your implementation will be evaluated using the Calico test module on ITU's Linux Server. The Calico test script named "**q2_calico_test.t**" is given with the question related documents/files. You can test your implementation using the below command before your submission:

```
python -m calico.cli q2_calico_test.t
```

The output of the Calico module of a successful implementation is given in Figure 5. For the evaluation, the point space statistics (coordinates of points and number of points) do not necessarily be in the output. But the last line (i.e., "Distance between the closest points: 5") of the output is necessary.

```
[[beyzaeken@ssh Beyza Eken Q2]$ ./q2 points10.txt
Points coordinates (x y):
65      70
32      44
120     78
22      98
354     290
1       122
36      65
65      36
87      78
44      256
Total number of points: 10
Distance between the closest points: 21.3776
```

Figure 3: The output for test file "points10.txt".

```
[[beyzaeken@ssh Beyza Eken Q2]$ ./q2 points20.txt
Points coordinates (x y):
2694    1833
1879    2186
2572    2529
2291    2899
2733    1659
1882    2257
706     97
302     384
2123    1088
134     2460
178     356
1686    2627
1661    664
1028    2622
699     209
2970    1117
1147    2195
1063    661
238     2503
2413    2169
Total number of points: 20
Distance between the closest points: 71.0634
```

Figure 4: The output of test file "points20.txt".

```
[[beyzaeken@ssh Beyza Eken Q2]$ python -m calico.cli q2_calico_test.t
init .................................... PASSED
build ................................... PASSED
case1 ................................... 1 / 1
case2 ................................... 1 / 1
case3 ................................... 1 / 1
Grade: 3 / 3
```

Figure 5: Output of Calico test.

Bibliography

[1] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006. 1, 2