**CSC IP (381): Project #11 <ArcChordDistance>(C++)**

**Student name: Jian Chen**

**Due Date: 05/12/2017**

**Submit Date: 05/10/2017**

**Algorithm Steps for the implementation for this project:**

step 0: - open all files.

- KChordLength <-- ask the user to type from console!!!!!!

- [numRows, numCols, minVal, maxVal, label, numPts] <- read from argv[1]

- dynamically allocate image array of size numRows by numCols

- dynamically allocate boundPtAry with size of numPts

- dynamically allocate chordAry with size of KChordLength // initiallied to 0.0

- index <-- 0

step 1: (x, y) <-- read from input

loadData // store x, y to boundPtAry[index]

step 2: index ++;

step 3: repeat step 1 and step 2 until the file is empty // check to make sure index == to numPt-1

step 4: P1 <-- 0

P2 <-- KChordLength-1

step 5: index <-- 0

currPt <-- P1 + 1

step 6: dist <-- computeDistance (P1, P2, currPt )

store dist to chordAry[index]

index ++

currPt ++

step 7: repeat step 6 while index < KChordLength

step 8: print chordAry to debugging file (argv[5])

step 9: maxIndex <-- findMaxDist // find the max of distances of all points in chordAry

// and returns that index

whichIndex <-- P1 + maxIndex

bountdPtAry[whichIndex]'s maxVotes ++

update bountdPtAry[whichIndex]'s maxDist if necessary

step 10: print bountPtAry from P1 to P2

step 11: Increment P1, and P2, and then

mod (P1, numPts) and mod (P2, numPts)

// so the computation will continue wrapped around the boundray

step 12: repeat step 5 to step 11 until P2 == (KChordLength / 2)

step 13: print the info (x, y, maxVotes, maxDist) of the entire boundaryAry to argv[5] // five pts per textline

step 14: compute LocalMaxima for all point in boundPtAry[index], index from 0 to numPts-1

step 15: compute isCorner for all point in boundPtAry[index], index from 0 to numPts-1

step 16: output the info (x, y, corner) of the entire boundaryAry to argv[3]

step 17: Call loadPtImage // load the boundary point onto the imgAry, i.e.,

// for each boundaryAry[i], put it corner value at Img(x, y)

step 18: prettyPrint imgAry to argv[4]

step 19: close all files