Project 7 (Java): You are to implement the Hough Transform for line detection algorithm.

****

Language: Java
Project points:10 pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):

-0 (10/10 pts): on time, 11/16/2021 Tuesday before midnight
+1 (11/10 pts): early submission, 11/12/2021, Friday before midnight
-1 (9/10 pts): 1 day late, 11/17/2021 Wednesday before midnight
-2 (8/10 pts): 2 days late, 11/18/2021 Thursday before midnight
(-10/10 pts): non submission, 11/18/2021 Thursday after midnight

*** Name your soft copy and hard copy files using the naming convention as given in the project submission requirement.
*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in **the same email attachments** with correct email subject as stated in the email requirement; otherwise, your submission will be rejected.
==========

You will be given 5 test image data:

img1 : contains only 1 point.
img2 : contains 2 points.
img3 : contains 3 points.
img4 : contains points form two colinear lines.
img5 : contains points form three colinear lines.

What to do as follows:
1) Implement your program based on the specs given below.
2) Run and debug your program on img1 until you see 1 sinusoid in Hough Space.
3) Run and debug your program on img2 until you see 2 sinusoids in Hough Space.
4) Run and debug your program on img3 until you see 3 sinusoids in Hough Space.
5) Run your program on img4, you should have multiple sinusoids what intersect at a point (or near-by) with 3 votes in Hough Space.
6) Run your program on img5, you should have multiple sinusoids what intersect at a point (or near-by) total with 6 votes in Hough Space.

*** Include in your hard copies:

- cover page
- source code
- outFile1 and outFile2 from the results of 2) in the above.
- outFile1 and outFile2 from the results of 3) in the above.
- outFile1 and outFile2 from the results of 4) in the above.
- outFile1 and outFile2 from the results of 5) in the above.
- outFile1 and outFile2 from the results of 6) in the above.

****

I. inFile(args[0]): a binary image with header

****

II. out puts: You will have two outFiles

1) outFile1 (args[1]): prettyPrint for visual
2) outFile2 (args[2]): The final result of your HoughAry with header information

****
III. Data structure:
****

- A HoughTransform class
    - (int) numRows
    - (int) numCols
    - (int) minVal
    - (int) maxVal
    - (int) HoughMinVal
    - (int) HoughMaxVal
    - (int) HoughDist // 2 times of the diagonal of the image
    - (int) HoughAngle // 180
    - (int) imgAry [][] // a 2D int array size of numRows by numCols; needs to dynamically allocate.
    - (int) HoughAry [][] // a 2D int array size of HoughDist by HoughAngle; needs to dynamically allocate.
    - (int) angleInDegree
    - (double) angleInRadians
    - (int) offSet // see lecture note.
  - methods:
    - constructor(...)
    - loadImage (…) // load imgAry from inFile
    - buildHoughSpace (...) // See algorithm steps below
    - polarDistance (point, angleInRadians) // on your own
      // use the polar distance formula is given in the Lecture Notes
    - determineMinMax (HoughAry) // on your own
      // read the entire HoughAry to determine HoughMinVal and HoughMaxVal
    - prettyPrint (…) // As in your previous projects
    - ary2File (HoughAry, outFile2) // output HoughAry to outFile2
    - add other methods and/or variables as needed.

****
IV. main (…)
****

Step 0: inFile ← open input file from args
outFile1, outFile2 ← open from args
numRows, numCols, minVal, maxVal ←- read from inFile
HoughAngle ← 180
HoughDist ←2 * (the diagonal of the input image)
imgAry ← dynamically allocate
HoughAry ← dynamically allocate HoughAry, size of
HoughDist by HoughAngle and initialize to zero

Step 1: loadImage (inFile)
Step 2: buildHoughSpace (...) // See algorithm below.
Step 3: prettyPrint (HoughAry, outFile1)
Step 4: determineMinMax (HoughAry)
Step 5: outFile2 ← HoughDist, HoughAngle, HoughMinVal, HoughMaxVal to outFile2
// as the header of Hough image
step 6: ary2File (HoughAry, outFile2) // output HoughAry to outFile2

Step 7: close all files

****
IV. buildHoughSpace (...)
****
Step 1: scan imgAry left to right and top to bottom
Using x for rows and y for column
Step 2: imgAry (x, y) ← next pixel
Step 3: if imgAry (x, y) > 0
computeSinusoid (x, y)
Step 4: repeat step 2 to step 3 until all pixels are processed

****
V. computeSinusoid (x, y)
****
Step 1: angleInDegree ← 0
Step 2: angleInRadians ← angleInDegree / 180.00 * pi
Step 3: dist ← polarDistance (x, y, angleInRadians)
Step 4: distInt ← (int) dist // cast dist from double to int
Step 5: HoughAry[distInt][angleInDegree]++
Step 6: angleInDegree ++
step 7: repeat step 2 to Step 6 while angleInDegree <= 179

****
VI. polarDistance (x, y, angleInRadians)
****
// Use the polar distance formula given in the Lecture Notes
/ Make sure the x & y coordinate need to convert to double in computation
// add offSet to the computation.