Name: Alif Rahi
Section: CSCI 381 - Computer Vision / Tues-Thurs 1:40-2:55pm
Project: 6
Due: April 19, 2023

```
*******************************
IV. main (…)
*******************************
Step 0:    inFile, outFile1 ← open from args []
           numRows, numCols, minVal, maxVal ← read from inFile
           HoughAngle ← 180
           HoughDist ←2 * (the diagonal of the input image)
           imgAry ← dynamically allocate
           CartesianHoughAry ← dynamically allocate and initialize to zero
           PolarHoughAry ← dynamically allocate and initialize to zero
           offSet ←  // See your lecture note.
Step 1: loadImage (inFile, imgAry)
        prettyPrint (imgAry, outFile1)
Step 2: buildHoughSpace (...)
Step 3: prettyPrint (CartesianHoughAry, outFile1) // with caption indicate it is Cartesian Hough space
        prettyPrint (PolarHoughAry, outFile1) // with caption indicate it is Polar Hough space
Step 4: close all files
```

```
package RahiA_Project6;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class RahiA_Project6_Main {
        static Scanner inFile;
        static BufferedWriter prettyPrintFile;
        static BufferedWriter debugFile;


        static class HoughTransform{
           int numRows, numCols, minVal, maxVal;
           int houghDist, houghAngle = 180, offSet;
           int[][] imgAry, CartesianHoughAry, PolarHoughAry;
           int angleInDegree;
           double angleInRadians;
```

```
        public HoughTransform(int numImgRows, int numImgCols,int imgMin,int imgMax)
throws IOException{
                        this.numRows = numImgRows;
                        this.numCols = numImgCols;
                        this.minVal = imgMin;
                        this.maxVal = imgMax;
                        imgAry = new int[numImgRows][numImgCols];
                        double diagLength = Math.sqrt((double)numImgRows*numImgRows +
numImgCols*numImgCols);
                        houghDist =(int)  Math.ceil(2* (  diagLength ));
            offSet = (int) Math.sqrt((double)numImgRows*numImgRows +
numImgCols*numImgCols);
            CartesianHoughAry = new int[houghDist][180];
            PolarHoughAry = new int[houghDist][180];
            zero2DAry(CartesianHoughAry);
            zero2DAry(PolarHoughAry);
        }
        void loadImage(){
            for(int i =0; i<numRows; i++){
                for(int j=0; j<numCols; j++){
                    imgAry[i][j] = inFile.nextInt();
                }
            }
        }
        void zero2DAry(int[][] ary){
            for(int i =0; i<ary.length; i++){
                for(int j=0; j<ary[i].length; j++){
                    ary[i][j] = 0;
                }
            }
        }
        void buildHoughSpace(){
            for(int i =0; i<numRows; i++){
                for(int j=0; j<numCols; j++){
                    if(imgAry[i][j] > 0) computeSinusoid(i,j);
                }
            }
        }
        void computeSinusoid(int i, int j){
            double angleInRadians;
            for(int angleInDegree =0; angleInDegree<180; angleInDegree++){
                angleInRadians = angleInDegree * (Math.PI / 180.00);
                int cartesianDist = (int) CartesianDistance(i, j, angleInRadians);
                int polarDist = (int) polarDistance(i,j,angleInRadians);

                CartesianHoughAry[cartesianDist][angleInDegree]++;
                PolarHoughAry[polarDist][angleInDegree]++;
            }
        }
```

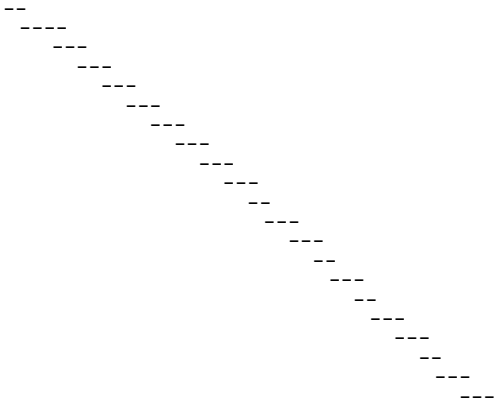
```
            double polarDistance(int x, int y, double theta){
               return  x * Math.cos(theta) + y * Math.sin(theta) + offSet;
            }

            double CartesianDistance(int x, int y, double theta){
                  double t = theta - Math.atan(y/x) - (Math.PI / 2);
                  double val = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
                  double dist = val * Math.cos(t) + offSet;
                  return dist;
            }

            void prettyPrint(int[][] ary) throws IOException{
               for(int i=0; i<houghDist; i++){
                  prettyPrintFile.write("\n");
                 for(int j=0; j<180; j++){
                    if(i< ary.length && j < ary[i].length && ary[i][j] > 0 ) prettyPrintFile.write("_ ");
                    else prettyPrintFile.write("  ");
                 }
               }
            }
         }
   public static void main(String[] args) throws IOException{
                  File imgFile = new File(args[0]);
                  File outFile = new File(args[1]);
                  prettyPrintFile = new BufferedWriter(new FileWriter(outFile));

                  int numImgRows, numImgCols,imgMin,imgMax;
         inFile = new Scanner(imgFile);
                  numImgRows = inFile.nextInt();
                  numImgCols = inFile.nextInt();
                  imgMin = inFile.nextInt();
                  imgMax = inFile.nextInt();
                  HoughTransform houghTransform = new
HoughTransform(numImgRows,numImgCols,imgMin,imgMax);
                  houghTransform.loadImage();
                  houghTransform.prettyPrint(houghTransform.imgAry);
                  prettyPrintFile.write("\n");

                  houghTransform.buildHoughSpace();
                  prettyPrintFile.write("Printing CartesianHoughSpace: \n");
                  houghTransform.prettyPrint(houghTransform.CartesianHoughAry);
                  prettyPrintFile.write("\n");
                  prettyPrintFile.write("Printing PolarHoughSpace: \n");
                  houghTransform.prettyPrint(houghTransform.PolarHoughAry);
                  prettyPrintFile.write("\n");


   }
}
```

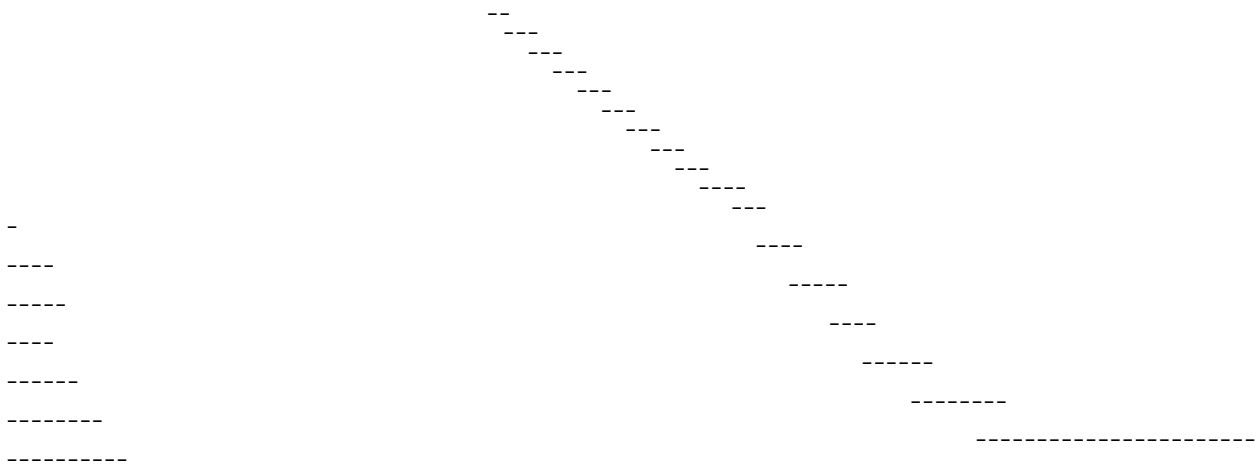
Outputs for all input files

img1pt.txt

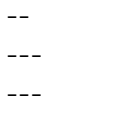
-

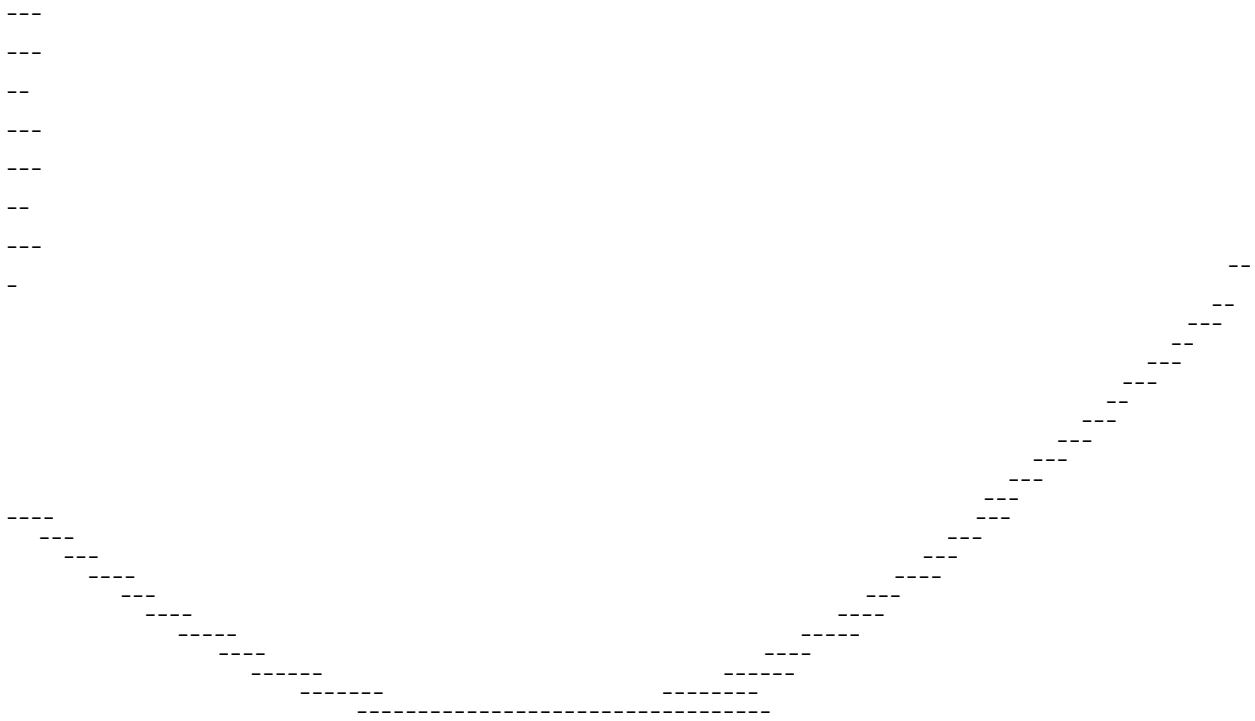
Printing CartesianHoughSpace:



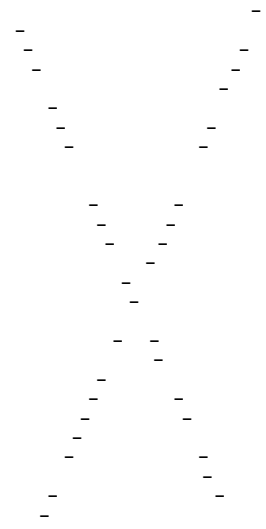


Printing PolarHoughSpace:

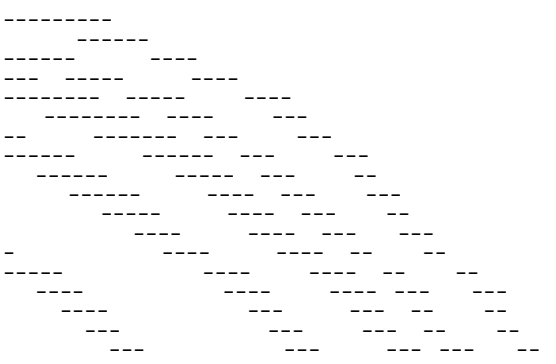


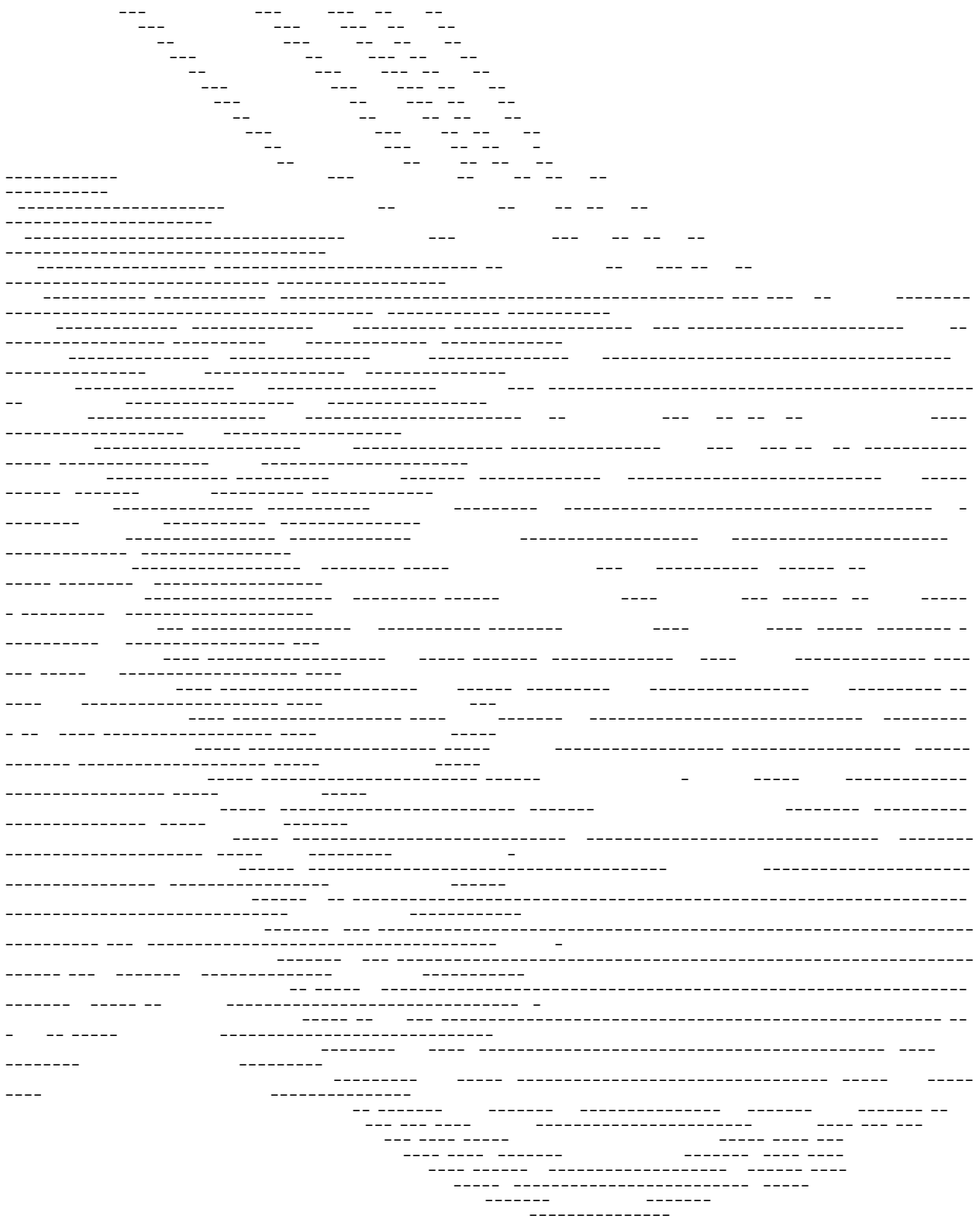


img2lines.txt



Printing CartesianHoughSpace:

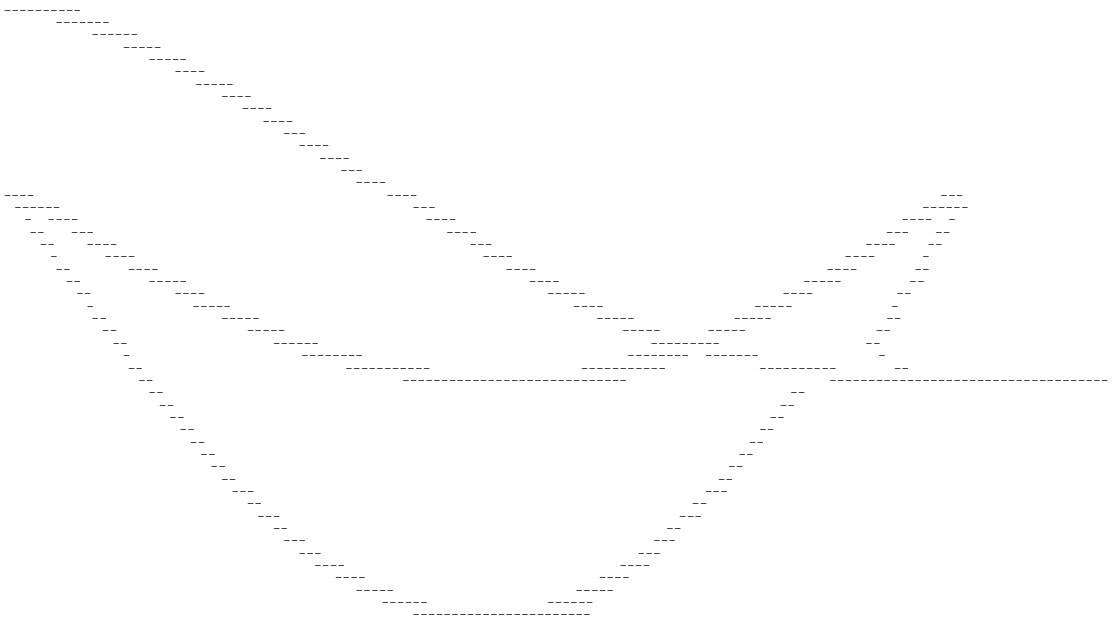




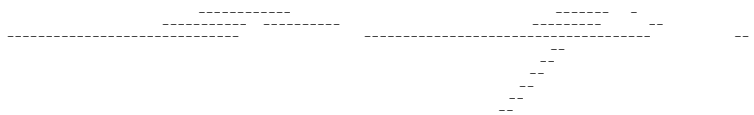
Printing PolarHoughSpace:

img3pt.txt

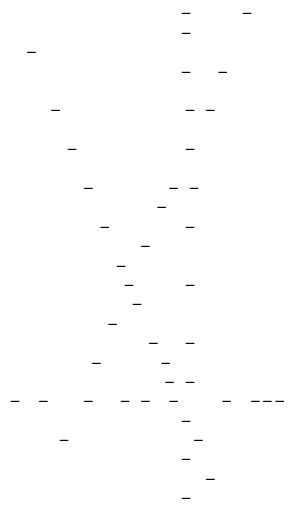
Printing CartesianHoughSpace:



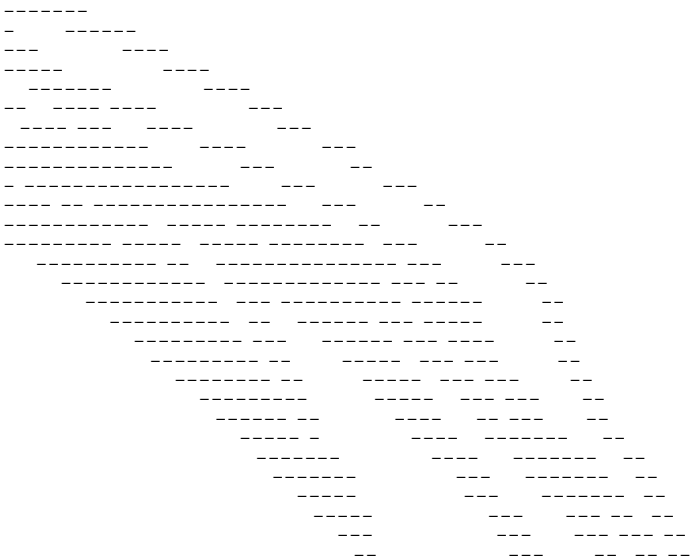
Printing PolarHoughSpace:



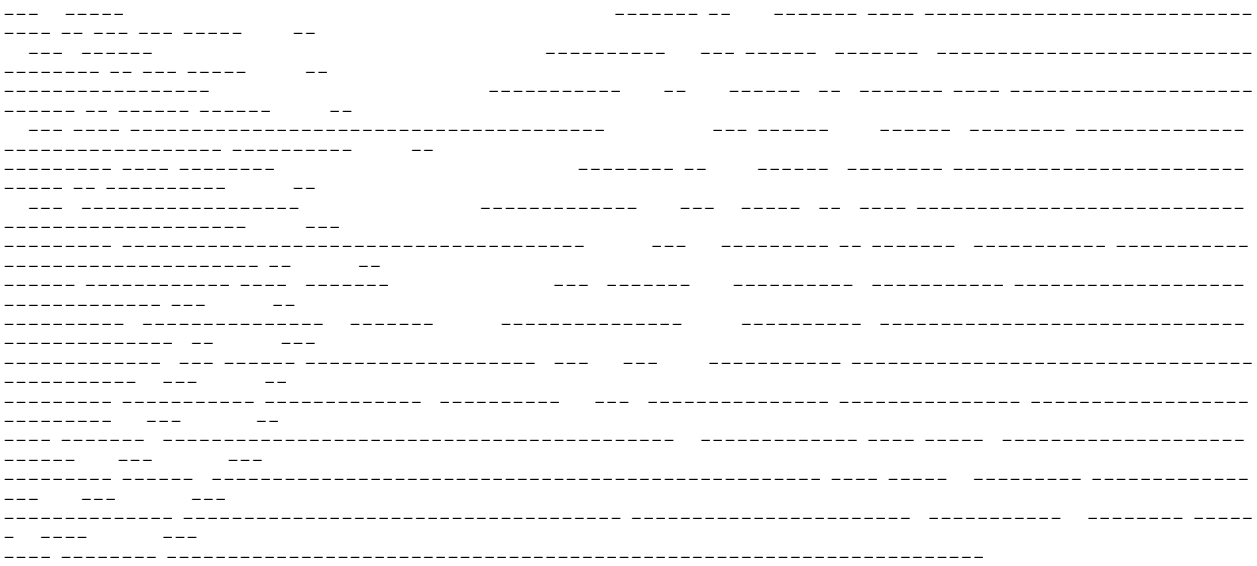
img5lines.txt



Printing CartesianHoughSpace:



Printing PolarHoughSpace:



img5pt.txt

Printing CartesianHoughSpace:

Printing PolarHoughSpace: