Student: Simon Kong

Project Due date: 9/16/2021

Algorithm Steps:

Main(args)

step 0: open inFile, maskFile

open all out files

thrVal ← get from args[2]

step 1: numRows, numCols, minVal, maxVal ← read from inFile

maskRows, maskCols, maskMin, maskMax ← read from maskFile

step 2: dynamically allocate all 1-D and 2-D arrays

step 3: loadMask (...)

step 4: loadImage (...)

step 5: mirrorFraming (...)

Step 6: imgReformat (mirrorFramedAry, minVal, maxVal, inputImg)

step 7: computeMedian (...)

step 8: imgReformat (medianAry, newMin, newMax, MedianOutImg)

step 9: threshold (medianAry, thrAry)

step 10: imgReformat (thrAry, newMin, newMax, MedianThrImg)

step 11: prettyPrint (thrAry, MedianPrettyPrint)

step 12: computeGauss (...)

step 13: imgReformat (GaussAry, newMin, newMax, GaussOutImg)

step 14: threshold (GaussAry, thrAry)

step 15: imgReformat (thrAry, newMin, newMax, GaussThrImg)

step 16: prettyPrint (thrAry, GaussPrettyPrint)

step 17: close all files

computeMedian()

step 0: newMin □ 9999; newMax ← 0

step 1: i ← 1

step 2: j ← 1

step 3: loadNeighbors (i, j, neighborAry)

step 4: sort (neighborAry)

step 5: medianAry [i,j] ← neighborAry[4]

step 6: if newMin > medianAry [i,j]

newMin ← medianAry [i,j]

if newMax < medianAry [i,j]

newMax ← medianAry [i,j]

step 7: j++

step 8: repeat step 3 to step 7 while j ← numCols

step 9: i++

step 10: repeat step 2 to step 9 while i ← numRows

computeGauss()

step 0: newMin ← 9999; newMax □ 0

step 1: i <- 1

step 2: j <- 1

step 3: GaussAry [i,j] ← convolution (i, j, mirrorFramedAry, maskAry)

step 4: if newMin > GaussAry [i,j]

newMin <- GaussAry [i,j]

if newMax < GaussAry [i,j]

newMax □ GaussAry [i,j]

step 5: j++

step 6: repeat step 3 to step 5 while j <= numCols

step 7: i++

step 8: repeat step 2 to step 6 while I <= numRows

imgReformat(inAry, newMin, newMax, outImg)

Step 1: OutImg ← output numRows, numCols, newMin, newMax

Step 2: str ← to_string(newMax)

Width ← length of str

Step 3: r ← 1

Step 4: c ← 1

Step 5: OutImg ← inAry[r][c]

Step 6: str ← to_string (inAry[r][c])

WW ← length of str

Step 7: OutImg ← one blank space

WW ++

Step 8: repeat step 7 while WW < Width

Step 9: c++

Step 10: repeat Step 5 to Step 9 while c <= numCols

Step 11: r++

Step 12: repeat Step 4 to Step 10 while r <= numRows

Threshold(ary1, ary2)

step 0: newMin ← 0

newMax ← 1

step 1: i ← 1

step 2: j ← 1

step 3: if ary1[i][j] >= thrVal

ary2[i][j] ← 1

else

ary2[i][j] ← 0

step 4: j++

step 5: repeat step 3 to step 4 while j < numCols+2

step 6: i++

step 7: repeat step 2 to step 6 while i < numRows+2

convulsion(I, j, ary, maskAry)

1. Initialize totalWeight, product, l, k
2. Start loop at top-left diagnol from center point and stop after 3 rows
3. Loop for 3 columns
4. Get sum of the products for values that have matching positions
5. Products ← sum of products
6. totalWeight = sum of maskAry values
7. divide products by totalweight

loadMask(scanner)

1) create 2d nested loop
2) first loop will loop for number of mask rows
3) second loop will loop for number of mask cols
4) use loop variables to read file values
5) save values into maskAry

loadImage(scanner)

1) create 2d nested loop
2) first loop will loop for number of rows
3) second loop will loop for number of cols
4) save value at loop variable + 1 to place values in center to be surrounded by padding

loadNeighbors(I, j)

1) index = 0 to count
2) start first loop at top left from center
3) nested loop to iterate through row
4) save every values in the 3x3 into 1D array by using index
5) index++ after every row

sort(arr)

1) get length of array
2) val = 0
3) created 2d nested loop that stops at length of array
4) for I = 0, for j = I + 1 //one space ahead of i
5) compare if arr[i] > arr[j]
6) if it is greater then swap values at the two index positions

mirrorFraming()

1) take first row of input and copy onto first row of array while skipping over first and last index (the corners)
2) take last row of input and copy onto last row of array while skipping over first and last index (the corners)
3) take first column of input and copy onto first column of array while skipping over first and last index (the corners)
4) take last column of input and copy onto last column of array while skipping over first and last index (the corners)
5) copy first input onto top left corner
6) copy last input in first row to top right corner
7) copy first input on last row to bottom left corner
8) copy last input to bottom right corner

Source Code

Main Class

```
public class Main {
    public static void main(String[] args) throws IOException {

            if(args.length != 10) {
                System.out.println("Invalid number of arguments.");
                System.exit(0);
            }

            try {
                Integer.parseInt(args[2]);
            }catch(Exception e){
                System.out.println("Invalid threshold.");
                System.exit(0);
            }

    //Initialize variables
            String inputFile = args[ 0 ];
            String maskFile = args[ 1 ];
            int thrVal = Integer.parseInt(args[2]);
            String inputImg = args[3];
            String medianOutImg = args[4];
            String medianThrImg = args[5];
            String medianPrettyPrint = args[6];
            String gaussOutImg = args[7];
            String gaussThrImg = args[8];
            String gaussPrettyPrint = args[9];

    //Initialize readers
            FileReader inputReader = null ;
            BufferedReader buffInReader = null ;
            Scanner input = null ;

            FileReader maskReader = null;
            BufferedReader buffMaskReader = null;
            Scanner mask = null;

    //Initialize writers
            FileWriter outputWriter = null ;
            BufferedWriter output = null ;
```

```
        try{
//          Open input
            inputReader = new FileReader( inputFile ) ;
            buffInReader = new BufferedReader( inputReader) ;
            input = new Scanner( buffInReader ) ;

            maskReader = new FileReader( maskFile );
            buffMaskReader = new BufferedReader( maskReader );
            mask = new Scanner( buffMaskReader );

            outputWriter = new FileWriter(inputImg);
            output = new BufferedWriter(outputWriter);

//          initialize variables
            int numRows = 0 ;
            int numCols = 0 ;
            int minVal = 0 ;
            int maxVal = 0 ;
            int maskRows = 0;
            int maskCols = 0;
            int maskMin = 0;
            int maskMax = 0;

            if( input.hasNextInt() ) numRows = input.nextInt() ;
            if( input.hasNextInt() ) numCols = input.nextInt() ;
            if( input.hasNextInt() ) minVal = input.nextInt() ;
            if( input.hasNextInt() ) maxVal = input.nextInt() ;

            if( mask.hasNextInt() ) maskRows = mask.nextInt();
            if( mask.hasNextInt() ) maskCols = mask.nextInt();
            if( mask.hasNextInt() ) maskMin = mask.nextInt();
            if( mask.hasNextInt() ) maskMax = mask.nextInt();

            imageProcess readObj = new imageProcess( numRows, numCols, minVal, maxVal, maskRows, maskCols, maskMin, maskMax, thrVal )

            readObj.loadMask(mask);
            readObj.loadImage(input);
            readObj.mirrorFraming();
            readObj.imgReformat(readObj.mirrorFramedAry, minVal, maxVal, output);
            output.close();
```

```
        outputWriter = new FileWriter(medianOutImg);
        output = new BufferedWriter(outputWriter);

        readObj.computeMedian();
        readObj.imgReformat(readObj.medianAry, readObj.newMin, readObj.newMax, output);

        output.close();

        outputWriter = new FileWriter(medianThrImg);
        output = new BufferedWriter(outputWriter);

        readObj.threshold(readObj.medianAry, readObj.thrAry);
        readObj.imgReformat(readObj.thrAry, readObj.newMin, readObj.newMax, output);

        output.close();

        outputWriter = new FileWriter(medianPrettyPrint);
        output = new BufferedWriter(outputWriter);
        readObj.prettyPrint(readObj.thrAry, output);

        output.close();

        outputWriter = new FileWriter(gaussOutImg);
        output = new BufferedWriter(outputWriter);

        readObj.computeGauss();
        readObj.imgReformat(readObj.gaussAry, readObj.newMin, readObj.newMax, output);

        output.close();

        outputWriter = new FileWriter(gaussThrImg);
        output = new BufferedWriter(outputWriter);

        readObj.threshold(readObj.gaussAry, readObj.thrAry);
        readObj.imgReformat(readObj.thrAry, readObj.newMin, readObj.newMax, output);

        output.close();


                readObj.threshold(readObj.gaussAry, readObj.thrAry);
                readObj.imgReformat(readObj.thrAry, readObj.newMin, readObj.newMax, output);

                output.close();

                outputWriter = new FileWriter(gaussPrettyPrint);
                output = new BufferedWriter(outputWriter);

                readObj.prettyPrint(readObj.thrAry, output);

            }finally {
                if( input != null ) input.close() ;
                if( mask != null ) mask.close();
                if( output != null ) output.close();
            }
        }
}
```

ImageProcess Class

Constructor()

```
public imageProcess(int rows, int cols, int min, int max, int mRows, int mCols, int mMin, int mMax, int thr){
    this.numRows = rows;
    this.numCols = cols;
    this.minVal = min;
    this.maxVal = max;
    this.thrVal = thr;

    this.maskRows = mRows;
    this.maskCols = mCols;
    this.maskMin = mMin;
    this.maskMax = mMax;

    this.thrVal = thr;

    this.mirrorFramedAry = new int[ this.numRows + 2 ][ this.numCols + 2];
    this.medianAry = new int[ this.numRows + 2 ][ this.numCols + 2];
    this.gaussAry = new int[ this.numRows + 2 ][ this.numCols + 2];
    this.thrAry = new int[ this.numRows + 2 ][ this.numCols + 2];
    this.maskAry = new int[ this.maskRows ][ this.maskCols];

    this.neighborAry = new int[9];
}
```

---

loadMask()

```
public void loadMask(Scanner maskFile) {
    for(int i = 0; i < this.maskRows; i++) {
        for(int j = 0; j < this.maskCols; j++) {
            if( maskFile.hasNextInt() ) this.maskAry[i][j] = maskFile.nextInt();
        }
    }
}
```

loadImage()

```
public void loadImage(Scanner inputFile) {
    for(int i = 0; i < this.numRows; i++) {
        for(int j = 0; j < this.numCols; j++) {
            if( inputFile.hasNextInt() ) this.mirrorFramedAry[i+1][j+1] = inputFile.nextInt();
        }
    }
}
```

loadNeighbors()

```
//  load neighbors for 3x3
    private void loadNeighbors(int i, int j) {
        int index = 0;

        for(int r = i - 1; r <= i + 1; r++) {
            for(int c = j - 1; c <= j + 1; c++) {
                this.neighborAry[index] = this.mirrorFramedAry[r][c];
                index++;
            }
        }

    }
```

Sort()

```
  public void sort(int[] arr) {
      int l = arr.length;
      int val = 0;

       for(int i = 0; i < l; i++) {
           for(int j = i + 1; j < l; j++) {
               if(this.neighborAry[i] > this.neighborAry[j]) {
                   val = this.neighborAry[i];
                   this.neighborAry[i] = this.neighborAry[j];
                   this.neighborAry[j] = val;
               }
           }
       }
  }
```

imgReformat()

```
public void imgReformat(int[][] inAry,int newMin,int newMax,BufferedWriter outImg) throws IOException {
    outImg.write( Integer.toString(this.numRows) + " " );
    outImg.write( Integer.toString(this.numCols) + " ");
    outImg.write( Integer.toString(newMin) + " " );
    outImg.write( Integer.toString(newMax) + "\n" );

    String str = Integer.toString(newMax);
    int width = str.length();
    int r = 1;
    int c = 1;

    while( r <= this.numRows ) {
        c = 1;
        while( c <= this.numCols ) {
            outImg.write( Integer.toString(inAry[r][c]) + " ");
            str = Integer.toString(inAry[r][c]);
            int ww = str.length();

            while( ww < width) {
                outImg.write(" ");
                ww++;
            }
            c++;
        }
        outImg.write("\n");
        r++;
    }
}
```

computeMedian()

```
public void computeMedian() {
    this.newMin = 9999;
    this.newMax = 0;
    int i = 1;
    int j = 1;

    while( i <= this.numRows ) {
        j = 1;
        while( j <= this.numCols ) {
            this.loadNeighbors(i, j);
            this.sort(this.neighborAry);
            this.medianAry[i][j] = this.neighborAry[4];
            if( this.newMin > this.medianAry[i][j] ) this.newMin = this.medianAry[i][j];
            if( this.newMax < medianAry[i][j] ) this.newMax = this.medianAry[i][j];
            j++;
        }
        i++;
    }

}
```

ComputeGauss()

```
public void computeGauss() {
    this.newMin = 9999;
    this.newMax = 0;
    int i = 1;
    int j = 1;
    while( i <= this.numRows ) {
        j = 1;
        while( j <= this.numCols ) {
            this.gaussAry[i][j] = this.convolution(i, j, this.mirrorFramedAry, this.maskAry);
            if(this.newMin > this.gaussAry[i][j]) this.newMin = this.gaussAry[i][j];
            if(this.newMax < this.gaussAry[i][j]) this.newMax = this.gaussAry[i][j];
            j++;
        }
        i++;
    }


}
```

Convolution()

```
private int convolution(int i,int j, int[][] ary, int[][] mask) {
    int totalWeight = 0;
    int product = 0;
    int l = 0;
    int k = 0;

    for(int r = i - 1; r <= i + 1; r++) {
        k = 0;
        for(int c = j - 1; c <= j + 1; c++) {
            product += ( mask[l][k] * ary[r][c] );
            totalWeight += mask[l][k];
            k++;
        }
        l++;
    }

    int weightAverage = product/totalWeight;
    return weightAverage;
}
```

Threshold

```
public void threshold(int[][] ary1, int[][] ary2) {
    this.newMin = 0;
    this.newMax = 1;
    int i = 1;
    int j = 1;

    while( i < this.numRows + 2 ) {
        j = 1;
        while( j < this.numCols + 2) {
            if(ary1[i][j] >= this.thrVal) ary2[i][j] = 1;
            else ary2[i][j] = 0;
            j++;
        }
        i++;
    }
}
```

prettyPrint

```
public void prettyPrint(int[][] inAry, BufferedWriter outFile) throws IOException {
    for(int i = 0; i < this.numRows; i++) {
        for( int j = 0; j < this.numCols; j++) {
            if(inAry[i][j] > 0) outFile.write( Integer.toString(inAry[i][j]) + " ");
            else outFile.write(". ");
        }
        outFile.write("\n");
    }
}
```

inputImg

```
46 46 1 63
1  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5
2  1  2  3  4  55 1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  43 4  5  1  2  3  4  5  1  2  3  4  5
3  1  2  3  4  5  1  2  3  4  45 51 2  3  4  5  1  2  3  4  5  1  2  48 4  5  1  2  43 4  5  1  2  3  4  5  11 2  3  4  5
4  1  2  3  4  5  1  2  3  4  55 51 2  3  4  5  1  2  3  4  5  1  2  48 4  5  1  2  43 4  5  1  2  3  4  5  1  2  3  4  5
5  1  22 3  4  5  1  2  43 4  5  1  2  3  4  5  1  2  3  4  5  1  2  38 4  5  1  2  43 4  35 1  2  3  4  5  1  2  3  4  5
6  1  2  3  4  5  1  2  3  44 5  1  2  3  4  5  1  2  3  4  5  1  2  41 4  5  1  2  3  44 5  1  2  3  4  5  1  2  3  4  5
7  1  2  3  4  5  1  2  3  44 5  1  2  3  4  5  8  2  3  4  5  1  2  38 4  5  1  12 3  44 5  1  2  3  4  5  1  2  31 4  5
8  1  2  3  4  5  1  2  53 4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  45 1  2  3  4  55 1  2  3  4  5
9  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  48 38 48 5  1  2  3  44 5  1  2  3  4  5  1  2  3  4  5
10 11 2  43 4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  48 4  48 48 48 1  2  43 4  5  1  2  3  4  5  1  2  3  4
1  1  2  3  4  5  11 2  3  4  5  1  2  3  4  5  1  2  3  4  48 33 4  4  41 48 48 2  3  44 5  1  2  3  4  5  1  2  3  4  5
2  1  2  3  4  45 51 2  3  4  5  1  2  3  4  5  1  2  3  48 48 48 4  8  48 48 48 48 3  4  5  19 2  3  4  5  1  2  3  14 5
3  1  2  3  4  55 51 2  3  4  5  1  2  3  4  5  1  2  4  8  4  8  8  4  4  4  8  8  8  4  5  1  2  3  4  5  1  2  3  44 5
4  21 22 23 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 22 20 18 18 16 13 4  5
5  1  21 3  4  5  1  2  3  4  5  1  2  3  4  5  1  48 48 48 58 10 41 48 42 34 48 45 48 48 5  1  2  3  4  5  1  2  3  4  5
6  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  48 48 58 48 58 48 48 48 48 48 48 48 4  48 48 1  2  3  4  5  1  2  3  4  55
7  1  2  3  14 5  1  2  3  4  5  1  2  3  4  48 48 38 41 42 43 41 42 63 4  48 48 46 48 48 48 48 2  3  4  51 1  2  3  4  5
8  1  2  3  4  5  1  2  3  4  5  1  2  3  48 41 38 44 43 48 45 48 44 48 48 48 48 58 48 48 4  4  48 3  4  5  1  2  3  4  5
9  1  2  3  4  15 1  12 3  4  5  1  2  48 48 60 48 60 41 48 41 48 48 61 62 48 48 43 48 8  7  48 48 48 4  5  1  2  3  4  5
10 1  2  3  4  5  1  2  3  4  5  1  48 48 48 5  48 48 48 3  48 48 48 48 48 6  48 48 47 48 8  48 48 48 48 5  1  12 3  4  5
1  1  52 3  4  5  1  12 3  4  5  48 41 58 48 43 48 48 48 48 28 38 48 43 48 48 41 48 8  48 45 28 48 42 48 48 1  2  3  4  5
2  1  2  3  4  5  1  2  3  4  48 48 58 48 48 48 40 48 47 48 42 48 41 48 42 48 52 48 4  8  5  48 48 48 38 48 48 48 3  4  5
3  21 22 23 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 22 20 18 18 16 13 4  5
4  1  2  3  4  5  1  2  48 48 48 48 48 48 4  58 48 48 38 58 58 58 38 38 58 48 58 58 28 24 14 48 48 48 48 48 48 48 48 4  5
5  1  2  48 41 48 42 48 43 8  48 60 58 48 48 38 41 42 48 43 48 46 48 45 48 40 48 4  3  48 30 48 48 48 8  42 48 48 38 48 48 28 48 48 4
6  1  2  3  4  5  1  2  48 58 58 48 38 48 48 38 48 63 48 63 48 48 48 48 4  48 48 48 8  4  48 48 48 48 48 18 48 48 48 48 4  5
7  1  2  3  4  5  1  2  3  48 48 8  48 38 42 48 48 18 48 48 48 48 63 48 48 48 48 48 48 8  8  48 48 48 5  48 48 48 3  4  5
8  1  2  3  4  5  13 2  3  4  48 48 62 63 55 48 48 48 4  7  8  48 48 48 54 48 58 48 48 4  4  8  48 48 48 48 48 2  3  4  5
9  1  2  3  4  5  1  2  3  4  5  48 48 58 48 38 48 48 28 8  48 48 48 48 48 48 18 48 48 6  4  8  4  48 48 48 1  2  3  4  5
10 1  2  3  4  5  1  2  3  4  5  1  48 48 58 48 3  48 58 48 40 48 58 18 48 48 48 41 48 42 48 8  4  48 48 5  1  2  3  4  5
1  21 22 23 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28 24 22 20 18 18 16 13 4  5
2  1  2  3  4  5  1  2  3  4  5  1  2  48 63 48 48 48 18 48 42 48 41 48 44 48 42 48 8  48 48 8  4  48 4  5  1  2  3  14 15
3  11 2  3  4  5  1  2  3  4  5  1  2  3  48 42 48 43 48 48 48 48 58 48 48 58 48 58 48 48 48 8  4  3  4  5  1  2  3  4  5
4  1  2  3  4  15 1  2  3  4  5  1  2  3  4  48 48 41 42 63 48 40 48 42 48 43 48 44 48 28 48 48 2  3  4  5  1  2  3  4  55
5  1  2  3  42 55 1  42 3  4  5  1  2  3  4  5  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  1  2  3  4  5  1  2  3  4  5
6  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  48 48 58 4  1  8  4  1  48 2  4  8  48 5  1  2  3  4  5  1  2  3  4  5
7  1  2  3  4  5  1  2  3  4  5  13 2  3  4  5  1  2  48 48 8  48 4  5  4  48 48 8  48 4  5  1  2  3  4  5  1  2  3  4  5
8  1  2  3  4  51 1  2  3  4  5  1  2  3  4  5  1  2  3  48 38 48 38 8  1  48 38 48 3  4  5  1  2  3  4  5  1  2  3  4  5
9  1  2  3  4  5  1  12 3  4  5  1  2  3  4  5  1  2  3  4  48 48 48 48 48 48 48 2  3  4  5  1  2  3  4  5  1  2  3  4  5
10 1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  48 48 18 48 48 1  2  3  4  5  1  2  3  4  5  1  2  3  4  5
1  1  2  3  44 5  1  2  3  4  5  1  12 3  4  5  1  2  3  4  5  1  48 48 48 5  1  2  3  4  5  1  2  3  4  55 1  2  3  4  5
2  1  2  3  48 5  1  2  3  4  55 51 12 3  4  5  1  2  3  4  5  1  2  48 4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5
3  1  2  3  4  45 51 2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  48 4  5  1  2  3  4  5  1  2  3  43 5  1  2  3  4  5
4  1  2  3  4  5  1  2  3  4  5  1  2  3  14 5  1  2  3  4  5  1  2  48 4  5  1  2  3  4  5  1  2  39 54 5  1  2  43 4  5
5  1  2  3  4  5  11 2  3  44 5  1  2  3  4  5  1  2  3  4  5  1  2  48 4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5
6  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5  1  2  3  4  5
```

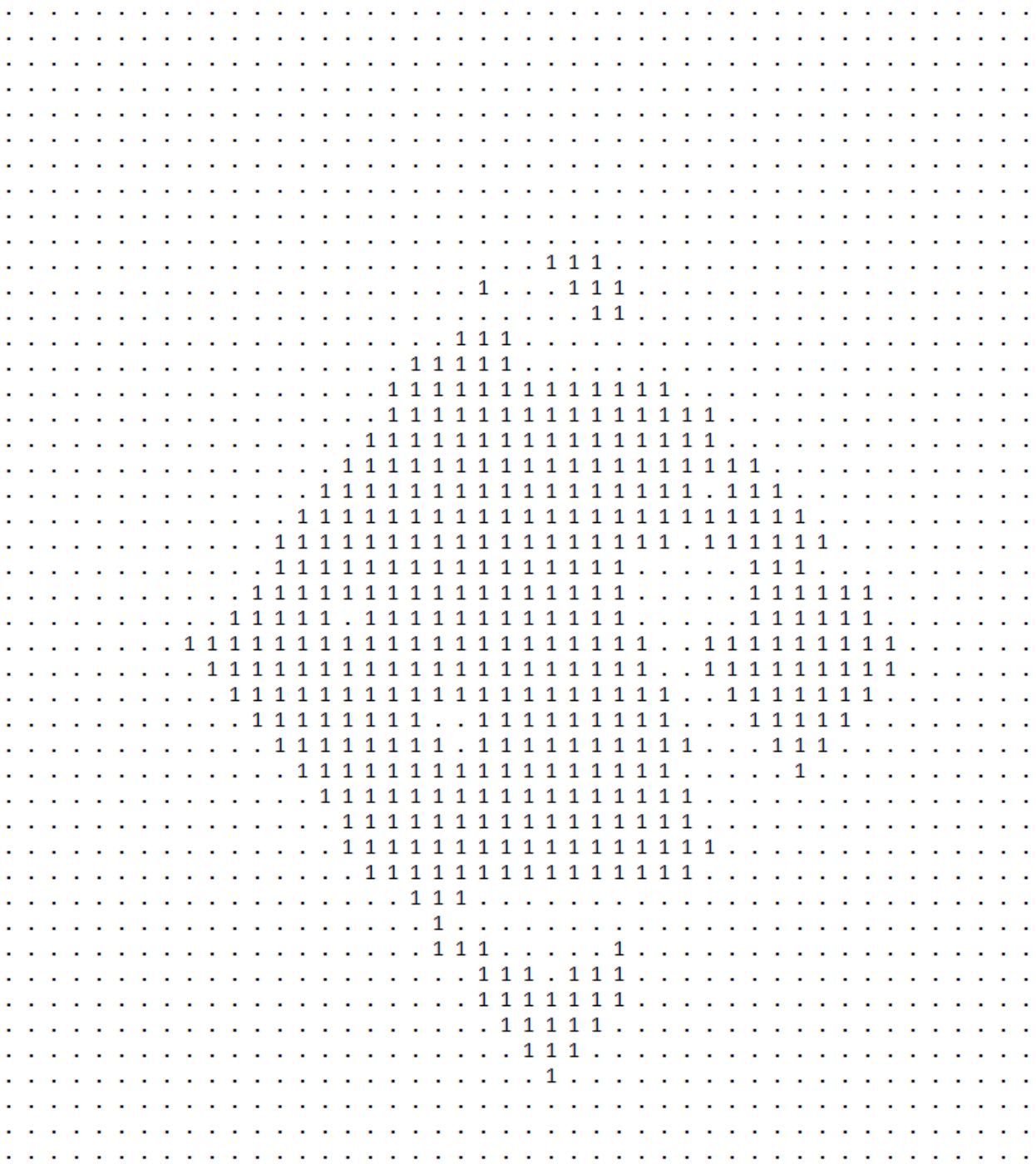
Median Output Image

```
46 46 1 58
1  1  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
2  2  2  3  4  4  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
3  2  2  3  4  4  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
4  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
5  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  4  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
6  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
7  2  2  3  4  4  2  2  3  5  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
9  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  5  4  38 38 5  2  2  3  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
9  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  5  33 41 48 48 2  2  3  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
2  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4  2  2  3  4  48 33 8  8  48 48 48 3  4  5  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
1  2  2  3  4  11 11 3  3  4  4  2  2  3  4  4  2  2  3  4  33 8  8  4  8  48 48 8  4  5  5  2  2  3  4  4  2  2  3  5  5  2  2  3  4  5
3  2  3  4  23 28 29 28 4  5  5  5  3  4  5  5  5  3  8  48 48 41 8  8  34 34 32 30 8  5  5  5  3  4  5  5  5  3  4  5  5  2  2  3  4  5
4  4  3  4  5  24 27 3  4  5  5  5  3  4  5  5  5  40 48 48 48 41 38 38 34 34 32 31 30 25 5  5  3  4  5  5  5  3  4  5  4  2  2  3  4  5
5  5  3  4  5  5  5  3  4  5  5  5  3  4  5  5  40 48 48 58 48 48 48 42 42 42 45 45 45 30 28 5  3  4  5  5  5  3  4  4  4  2  2  3  4  5
5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 48 43 48 48 48 48 48 48 48 48 48 2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
6  2  2  3  4  4  2  2  3  4  4  2  2  3  4  41 44 44 44 45 48 45 48 48 48 48 48 48 48 48 48 4  3  3  4  4  2  2  3  4  4  2  2  3  4  5
7  2  2  3  4  4  2  2  3  4  4  2  2  3  48 48 48 43 43 43 45 44 48 48 48 48 48 48 48 48 48 48 4  4  4  2  2  3  4  4  2  2  3  4  5
8  2  2  3  4  4  2  2  3  4  4  2  2  48 48 48 48 48 48 45 48 48 48 48 48 48 48 48 48 48 47 8  48 48 48 5  4  2  2  3  4  4  2  2  3  4  5
1  2  2  3  4  4  5  3  4  4  4  5  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 45 45 48 48 48 48 48 5  2  2  3  4  4  2  2  3  4  5
1  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8  45 48 48 48 48 48 12 3  4  4  4  2  2  3  4  5
2  2  3  4  5  5  5  3  4  5  32 41 48 48 48 43 48 48 48 48 48 42 43 42 43 42 48 32 30 25 28 28 42 42 42 38 18 13 4  4  4  2  2  3  5  5
3  2  3  4  5  5  5  3  29 32 48 48 48 48 48 40 48 48 48 58 58 48 41 41 42 48 48 32 28 24 25 28 48 48 48 48 48 48 48 13 5  4  2  2  3  5  5
4  3  3  22 24 27 28 31 31 43 48 48 48 48 38 40 42 48 48 58 58 48 45 41 40 48 40 31 28 28 28 30 48 48 42 42 48 48 38 13 5  5  3  4  5  5
4  2  2  3  5  5  5  42 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 24 24 48 48 48 48 48 48 48 48 48 48 38 5  5  3  4  4  5
5  2  2  3  5  5  5  3  43 48 48 48 48 48 48 48 42 48 48 48 48 48 48 48 48 48 48 48 48 48 8  8  48 48 48 48 48 48 48 48 48 5  5  5  3  4  4  5
6  2  2  3  4  4  2  2  3  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8  8  48 48 48 48 48 48 48 4  4  4  2  2  3  4  5
7  2  2  3  4  4  2  2  3  4  48 48 48 48 48 48 48 28 28 48 48 48 48 48 48 48 48 8  8  8  48 48 48 48 48 3  3  4  4  2  2  3  4  5
8  2  2  3  4  4  2  2  3  4  5  48 48 55 48 48 48 48 48 28 48 48 48 48 48 48 48 48 48 42 8  8  8  48 48 48 2  2  3  4  4  5  3  3  4  5
9  2  3  4  5  5  5  3  4  5  5  34 48 48 48 40 48 48 48 48 48 48 48 48 48 48 48 48 41 41 41 30 25 8  22 24 48 18 5  3  4  4  4  2  2  3  4  5
2  2  3  4  5  5  5  3  4  5  5  5  34 48 48 48 48 48 48 48 48 48 48 48 48 41 44 44 42 41 41 42 28 24 22 22 20 5  5  3  4  5  4  2  2  3  4  5
2  2  3  4  5  5  5  3  4  5  5  5  3  35 42 48 48 48 48 48 48 48 48 44 48 44 48 42 48 48 28 24 8  4  5  5  5  3  4  5  4  2  2  3  4  5
3  2  2  3  4  4  2  2  3  4  4  2  2  3  48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8  4  4  4  4  2  2  3  4  4  2  2  3  4  5
4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  42 42 42 42 48 48 48 42 48 43 48 44 48 44 48 28 4  3  3  4  4  2  2  3  4  4  2  2  3  4  5
4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  5  41 42 42 4  4  4  4  4  4  4  4  8  8  5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  4  48 8  4  4  4  4  4  4  4  4  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
6  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  48 48 48 8  8  4  5  38 48 8  8  5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
7  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  38 48 48 48 8  48 48 48 38 4  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
8  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  48 48 48 48 48 48 48 48 3  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
1  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5  48 48 48 48 48 48 2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
1  2  2  3  5  5  2  2  3  4  4  5  3  4  4  4  2  2  3  4  4  5  48 48 48 5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
1  2  2  3  5  5  2  2  3  4  4  5  3  4  4  4  2  2  3  4  4  2  2  48 5  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
2  2  2  3  4  5  2  2  3  4  4  5  3  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  5
3  2  2  3  4  5  5  3  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  5  5  2  2  3  4  4  2  2  3  4  5
4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  4  5  4  2  2  3  4  4  2  2  3  5  4  2  2  3  4  4  2  2  3  4  5
5  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  4  2  2  3  4  5
```

Median Threshold Image

```
46 46 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Median Pretty Print



Gaussian Output Image

```
46 46 1 55
1  1  2  3  7  10 5  2  3  4  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  3  2  4  8  6  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
1  1  2  3  10 16 8  2  3  6  11 11 5  3  4  3  2  2  3  4  3  2  4  8  6  3  2  9  18 11 3  2  2  3  4  4  3  2  3  4  3  2  2  3  4  4
2  1  2  3  7  10 5  2  3  12 29 29 11 3  4  3  2  2  3  4  3  2  10 19 12 3  2  12 23 14 3  2  2  3  4  5  4  3  3  4  3  2  2  3  4  4
3  3  4  4  4  3  2  4  8  15 30 29 11 3  4  3  2  2  3  4  3  2  12 24 14 3  2  12 23 15 7  4  2  3  4  4  3  2  3  4  3  2  2  3  4  4
4  4  7  5  4  3  2  7  15 17 15 11 5  3  4  3  2  2  3  4  3  2  11 22 13 3  2  9  20 20 13 6  2  3  4  3  2  2  3  4  3  2  2  3  4  4
4  3  4  4  4  3  2  4  15 21 11 2  2  3  4  4  3  2  3  4  3  2  11 21 13 3  2  5  16 23 15 4  2  3  4  3  2  3  6  5  3  2  2  3  4  4
5  2  2  3  4  3  2  5  16 22 11 2  2  3  4  4  4  2  3  4  3  2  8  16 10 3  3  4  11 21 16 4  2  3  7  10 5  5  10 7  3  2  2  3  4  4
6  3  2  3  4  3  2  8  18 15 6  2  2  3  4  4  3  2  3  4  3  5  12 17 13 6  2  3  8  19 18 7  2  3  10 16 8  3  6  5  3  2  2  3  4  4
7  4  5  8  6  3  2  5  9  7  3  2  2  3  4  3  2  2  3  4  6  14 23 31 30 17 4  4  13 19 13 4  2  3  7  10 5  2  3  4  3  2  2  3  4  4
7  5  8  13 9  4  3  2  3  4  3  2  2  3  4  3  2  2  3  6  17 23 24 32 41 33 16 9  18 19 8  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
3  3  5  8  9  13 13 6  3  4  3  2  2  3  4  3  2  2  5  17 32 30 16 20 38 44 33 19 15 16 9  4  3  3  4  3  2  2  3  5  4  2  2  3  4  4
1  1  2  3  12 30 30 12 3  4  3  2  2  3  4  3  2  2  8  23 33 27 12 13 28 36 34 23 12 9  8  6  4  3  4  3  2  2  6  11 7  2  2  3  4  4
3  5  7  8  17 36 36 18 9  10 10 10 10 10 11 12 12 13 20 27 29 25 18 16 20 22 23 20 14 10 10 10 8  7  8  7  6  5  10 15 9  2  2  3  5  5
5  10 14 14 17 24 24 18 16 17 17 18 18 18 19 20 24 34 42 44 41 33 31 31 29 28 28 29 28 24 17 14 13 12 12 11 9  8  9  10 6  4  5  6  6  6
5  8  11 10 9  9  8  8  9  10 10 10 10 10 11 15 26 43 51 53 49 39 40 43 41 40 41 40 37 34 23 10 7  7  8  7  6  5  5  7  10 9  9  7  5  5
4  3  4  4  5  4  2  2  3  4  3  2  2  3  6  17 33 44 48 49 47 43 44 45 41 43 46 41 36 39 33 16 4  3  6  9  5  2  3  10 16 10 5  4  4  4
5  2  2  4  6  5  2  2  3  4  3  2  2  5  17 32 42 43 44 45 46 45 47 45 38 42 48 46 42 42 36 25 13 5  9  15 8  2  3  7  10 5  2  3  4  4
6  3  2  3  5  5  3  3  3  4  3  2  4  16 33 43 45 44 44 44 44 45 47 48 46 46 48 49 46 34 23 26 28 17 9  9  5  2  3  4  3  2  2  4  4  4
7  3  2  3  5  6  4  4  4  4  3  5  16 33 43 44 46 48 44 40 43 46 49 52 50 44 45 47 42 27 19 32 42 34 17 6  2  3  3  4  3  2  3  5  5  4
5  6  8  6  4  5  4  4  4  4  6  16 33 45 43 38 43 49 42 34 38 45 47 49 45 37 41 44 39 32 28 37 46 44 34 17 6  4  4  4  4  3  3  4  4  4
2  8  14 9  4  3  3  4  4  6  17 33 45 50 45 40 43 47 45 39 38 42 45 45 44 41 44 39 29 29 31 37 44 45 43 33 19 11 6  4  5  4  3  3  5  5
3  8  13 11 9  9  9  10 10 16 30 42 46 46 45 43 44 47 50 49 45 44 43 43 43 43 44 35 22 19 25 34 40 39 38 36 31 22 10 4  4  3  2  4  7  6
5  9  12 13 14 15 15 18 25 31 37 41 42 39 36 40 44 47 52 55 54 49 42 40 41 42 42 37 27 21 23 32 36 34 32 32 32 29 18 8  3  2  2  4  7  6
4  5  9  15 19 19 19 25 34 38 42 46 46 39 34 40 45 45 48 53 55 50 43 42 45 45 44 36 28 26 28 37 42 39 35 37 40 39 32 20 13 11 12 12 8  5
4  2  7  18 24 24 23 29 36 39 46 52 50 45 41 41 45 46 47 49 50 48 46 43 41 44 43 29 20 25 35 43 48 43 35 37 45 46 40 30 23 20 22 20 9  4
4  2  4  10 14 14 12 18 31 43 47 45 44 45 44 43 44 46 49 51 49 48 49 43 36 41 44 34 21 20 32 44 48 42 33 34 43 44 32 20 13 11 12 11 6  4
5  2  2  3  4  4  3  5  17 35 42 38 43 47 46 46 44 39 38 39 41 47 51 47 43 46 49 46 32 16 19 35 45 42 35 38 43 33 17 6  4  3  2  3  4  4
6  3  2  3  4  5  5  3  5  17 31 42 50 54 50 47 45 37 25 21 30 43 49 49 49 48 46 47 37 16 8  19 35 42 42 42 33 16 5  4  5  4  3  3  4  4
7  3  2  3  4  4  3  2  3  6  17 34 49 54 51 43 41 40 31 25 34 45 47 45 46 45 41 44 39 23 13 11 21 39 45 33 16 4  3  4  4  4  3  3  4  4
7  6  7  8  9  9  8  8  9  10 13 24 39 47 47 39 34 42 47 45 45 48 45 39 41 42 39 40 39 33 26 16 18 33 35 21 8  5  5  4  3  3  4  4  5  5
5  9  12 13 14 15 15 15 16 17 17 20 29 39 44 41 39 44 50 52 51 47 43 39 39 40 38 35 33 34 31 21 20 26 23 13 9  8  7  7  5  3  3  4  6  6
3  6  7  8  9  9  8  8  9  10 10 10 15 32 45 46 45 43 42 46 48 47 46 44 44 44 43 39 34 37 35 19 15 19 13 7  6  5  6  8  7  3  2  3  5  5
3  4  3  3  4  5  2  2  3  4  3  2  4  17 35 44 46 43 43 47 47 47 48 47 48 49 49 47 42 43 39 21 9  9  6  3  2  2  3  9  11 6  2  3  4  4
3  3  2  5  13 14 9  7  5  4  3  2  2  5  17 30 35 33 36 40 37 35 36 36 36 37 37 37 34 32 31 23 8  3  4  3  2  2  3  10 16 8  2  3  4  4
4  2  2  7  20 22 14 12 8  4  3  2  2  3  6  12 17 21 26 26 18 13 14 14 16 19 16 14 16 18 16 10 4  3  4  3  2  2  3  7  10 5  2  3  4  4
4  2  2  5  11 12 7  7  5  4  4  3  2  3  4  3  8  22 35 31 16 9  7  4  11 22 17 10 16 18 9  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
5  2  2  3  6  9  5  2  3  4  5  5  3  3  4  3  5  16 34 38 28 25 17 6  14 33 33 23 21 15 6  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
6  3  2  3  9  15 8  3  3  4  4  3  2  3  4  3  2  4  16 30 37 40 32 20 23 38 40 27 14 6  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
7  3  2  3  6  9  6  4  4  4  3  2  2  3  4  3  2  2  5  16 33 44 42 35 37 41 32 15 5  4  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
5  2  2  5  9  6  2  3  3  4  3  2  3  3  4  3  2  2  3  6  17 34 41 40 41 33 16 4  3  4  3  2  2  3  7  10 5  2  3  7  10 5  2  3  4  4
2  1  2  10 19 11 2  2  3  7  13 13 8  4  4  3  2  2  3  4  6  16 31 38 32 17 4  2  3  4  3  2  2  3  10 16 8  2  3  10 16 8  2  3  4  4
1  1  2  11 22 19 11 5  3  10 22 22 12 4  4  3  2  2  3  4  3  5  19 31 20 6  2  2  3  4  3  2  2  5  12 12 5  2  3  7  10 5  2  3  4  4
2  1  2  5  14 22 19 8  3  7  13 12 6  4  5  4  2  2  3  4  3  2  13 25 15 3  2  2  3  4  3  2  4  15 22 11 2  4  8  6  3  2  3  6  5  4
3  2  2  3  6  12 12 5  5  9  6  2  2  4  6  5  2  2  3  4  3  2  13 25 15 3  2  2  3  4  3  2  6  20 25 12 2  7  13 9  3  2  5  10 7  4
4  2  2  3  4  5  4  3  8  14 8  2  2  3  5  4  2  2  3  4  3  2  10 19 12 3  2  2  3  4  3  2  4  10 12 6  2  4  8  6  3  2  3  6  5  4
4  2  2  3  4  4  3  2  5  9  6  2  2  3  4  3  2  2  3  4  3  2  4  8  6  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  3  2  2  3  4  4
```

Gaussian Threshold Image

```
46 46 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Gaussion Pretty Print