Project 2 (Java): You are to implement the three image enhancement methods taught in class: (1)3x3 median filter, and (2) 3x3 2D-Gaussian filter.

****

Project points: 10 pts

Language: Java

Due Date: Soft copy (*.zip) and hard copies (*.pdf):

-0 (10/10 pts): on time, 9/16/2021 Thursday before midnight

+1 (11/10 pts): early submission, 9/12/2021, Sunday before midnight

-1 (9/10 pts): 1 day late, 9/17/2021 Friday before midnight

-2 (8/10 pts): 2 days late, 9/18/2021 Saturday before midnight

(-10/10 pts): non submission, 9/18/2021 Saturday 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.

****

Include in your hard copy *.pdf file as follows:

- Cover page
- Source code
- inputImg file
- MedianOutImg file
- MedianThrImg file
- MedianPrettyPrint file
- GaussOutImg file
- GaussThrImg file
- GaussPrettyPrint file

****

I. Input files:

a) inFile (args[0]): A txt file representing a grey-scale image with image header.

b) maskFile (args[1]): a mask for convolution, with the following format:

MaskRows MaskCols MaskMin MaskMax,
follow by MaskRows by MaskCols of pixel values
For example, a 3 by 3 mask may be

```
3 3 1 4
1 2 1
2 4 2
1 2 1
```

c) a threshold value (args[2]) // USE 40

****

II. Output files:

1) inputImg (args[3]): the input image after reformatting.
2) MedianOutImg(args[4]): The image of the result of 3x3 median filter, after reformatting.
3) MedianThrImg(args[5]): The threshold result of 3x3 median filter, after reformatting.
4) MedianPrettyPring(args[6]): The pretty print of the threshold result of median filter
5) GaussOutImg(args[7]): The image of the result of 3x3 Gaussian filter, after reformatting.
6) GaussThrImg(args[8]): The threshold result of 3x3 Gaussian filter, after reformatting.
7) GaussPrettyPring(args[9]): The pretty print of the threshold result of Gaussian filter

****

III. Data structure:

****

- imageProcessing class
  - (int) numRows
  - (int) numCols (int)
  - (int) minVal (int)

```
- (int) maxVal
- (int) maskRows
- (int) maskCols
- (int) maskMin
- (int) maskMax
- (int) newMin
- (int) newMax)
- (int) thrVal  // from args[]

- (int) mirrorFramedAry [][] // a 2D array, dynamically allocate
                  //at run time of size numRows + 2 by numCols + 2.
- (int) medianAry [][]// a 2D array, dynamically allocate at run time
                            // of size numRows + 2 by numCols + 2.
- (int) GaussAry [][]// a 2D array, dynamically allocate at run time
                            // of size numRows + 2 by numCols + 2.
- (int) thrAry [][]// a 2D array, dynamically allocate at run time
                            // of size numRows + 2 by numCols + 2.
                            // to hold the threshold result.
- (int) maskAry [][]
          // a 2D Gaussian mask of size maskRows by maskCols,
          // used in the convolution
- (int) neighborAry [9]  // 1-D array to hold the 3x3 pixels

methods:

- threshold (file1, file2)// see algorithm below.
- imgReformat (...) // see algorithm below.
 - mirrorFraming (...) // On your own. The algorithm of Mirror framing was taught in class
 - loadImage (...) // On your own.  Read from input file and load onto mirrorFramedAry begin at [1][1].
 - loadMask (...)// On your own. Load the mask into the maskAry.
 - loadNeighbors(...) // On your own. Load the 3 x 3 neighbors of mirrorFramedAry (i,j) into neighborAry,
                  // using 2 loops;  do NOT write 9 assignments.
 - sort (neighborAry) // Use any sorting algorithm.  On your own.
 - computeMedian (...) // process the entire ary, keep track of newMin and newMax. See algorithm below.
 - computeGauss (...) // process the entire ary, keep track of newMin and newMax. See algorithm below.
 - (int) convolution (...) // As taught in class. Compute the convolution using the given mask
          // onto the pixel's maskRows by maskCols neighborhood.  On your own.
 - imgReformat (…) // See algorithm below.
- prettyPrint (inAry, outFile) // print without the frames.
                            // if ary[i][j] > 0
                                   outFile ← ary[i][j] follows by one blank space
                              else
                                   outFile ← "." follows by one blank space
```

****

IV. Main(...)

****

```
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 (...) // load maskFile onto maskAry
step 4: loadImage (...) // load inFile to mirrorFramedAry
step 5: mirrorFraming (...)
Step 6: imgReformat (mirrorFramedAry, minVal, maxVal, inputImg)
step 7: computeMedian (...) // see algorithm below
step 8: imgReformat (medianAry, newMin, newMax, MedianOutImg)
step 9: threshold (medianAry, thrAry) // see algorithm below
step 10: imgReformat (thrAry, newMin, newMax, MedianThrImg)
step 11: prettyPrint (thrAry, MedianPrettyPrint)
step 12: computeGauss (...) // see algorithm below
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
```

```
**************************************
V. computeMedian (...) // process the entire ary, keep track of newMin and newMax
**************************************
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
```

```
**************************************
VI. computeGauss (...) // process the entire ary, keep track of newMin and newMax
**************************************
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
```

```
******************************
VIII. imgReformat (inAry, newMin, newMax, OutImg)
******************************
Step 1: OutImg ← output numRows, numCols, newMin, newMax
Step 2: str ← to_string(newMax)  // a method in C++ string class
            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
```

```
******************************
VII. 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
```