Project 3 (Java) : Implementation of the four basic Morphology Operations.

A) You will be given four (4) image files: Img1, Img2, Img3, and Img4; and you will be given two (2) structuring elements: Elem1 and Elem2.
- Test1: to verify the correctness of your program, you will run your program using Img1 with Elem1.
- Test2: to verify the correctness of your program, you will run your program using Img2 with Elem2.

B) You are to design two (2) structuring elements: Elem3 and Elem4 for Img3 and Img4.
To design Elem3, make observation to see what are objects in Img3. The goal is to extract object(s) from Img3.
To design Elem4, make observation to see what are objects in Img4. The goal is to extract object(s) from Img4.

- Test3: to verify the correctness of your design, you will run your program using Img3 with Elem3.
- Test4: to verify the correctness of your design, you will run your program using Img4 with Elem4.

Your hard copies include:
- cover sheet
- program source code
- print all output files for test1
- print all output files for test2
- print all output files for test3
- print all output files for test4

****
Language: Java
Project points: 10 pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):
-0 2/28/2021 Sunday before midnight
-1 for 1 day late: 3/1/2021 Monday before midnight
-2 for 2 days late: 3/2/2021 Tuesday before midnight
-10/10: 3/2/2021 Tuesday after midnight
-5/10: does not pass compilation
0/10: program produces no output
0/10: did not submit hard copy.

*** Follow "Project Submission Requirement" to submit your project.

****
I. Inputs: There are two input files.
****
a) InFile1 (args[0]): a txt file representing a binary image with header.

b) InFile2 (args[1]): a txt file representing a binary image of a structuring element with header and the origin of the structuring element. The format of the structuring element is as follows:
1st text line is the header; the 2nd text line is the position (w.r.t. index) of the origin of the structuring element then follows by the rows and column of the structuring element.
For example:

```
5 5 0 1 // 5 rows, 5 columns, min is 0, max is 1: 2-D structuring element
2 2     // origin is at row index 2 and column index 2.
0 0 1 0 0
0 0 1 0 0
1 1 1 1 1
0 0 1 0 0
0 0 1 0 0
```

** Note: when a structure element contains zeros, only those 1's to be used in the matching in the erosion!

Another example:

```
3 3 1 1 // 3 rows, 3 columns, min is 1, max is 1: 2-D structuring element
1 1     // origin is at row index 1 and column index 1.
1 1 1
1 1 1
1 1 1
```

Another example:

```
1 5 1 1 // 1 rows, 5 columns, min is 1, max is 1: 1-D structuring element
0 2     // origin is at row index 0 and column index 2.
1 1 1 1 1
```

****

II. Outputs: (All of the following output files need to be included in your hard copies!)

- dilateOutFile (args[2]): the result of dilation image with header, without framed boarders.
- erodeOutFile (args[3]): the result of erosion image with header, the same dimension as imgFile
- closingOutFile (args[4]): the result of closing image with header, the same dimension as imgFile
- openingOutFile (args[5]): the result of opening image with header, the same dimension as imgFile
- prettyPrintFile (args[6]): pretty print which are stated in the algorithm steps

*** Note: When you run your program, please name your output files as given in the above.
*** NO HARD coded file names in the program, -2 points if you hard code file name in this project!!!

****

III. Data structure:

****

- numImgRows (int)
- numImgCols (int)
- imgMin (int)
- imgMax (int)

- numStructRows (int)
- numStructCols (int)
- structMin (int)
- structMax (int)
- rowOrigin (int)
- colOrigin (int)

- rowFrameSize (int) // set to numStructRows / 2
- colFrameSize (int) // set to numStructCols / 2

- extraRows (int) // set to rowFrameSize * 2
- extraCols (int) // set to colFrameSize * 2

- zeroFramedAry (int **) // a 2D array of size (numImgRows + extraRows) by (numImgCols + extraCols)
  // for the input image, needs to dynamically allocate at run time.
- morphAry (int **) // Same size as zeroFramedAry.
- tempAry (int **) // Same size as zeroFramedAry.
  // tempAry is to be used as the intermediate result in opening and closing operations.

- structAry (int **) //a 2D array of size numStructRows by numStructCols,
  // needs to dynamically allocate at run time.

Methods:

- loadImg (imgFile, zeroFramedAry) // On your own!
  // load imgFile to zeroFramedAry, begins at (rowOrigin, colOrigin) and ends at ??
- loadstruct (structFile, structAry) // On your own!
  // load structFile to structAry.
- zero2DAry (…) // set a given 2D array to zero.
- ComputeDilation (...) // process every pixel of the entire ary // see algorithm below.
- ComputeErosion (…)// process every pixel of the entire ary // see algorithm below.
- ComputeOpening (...)// see algorithm below.
- ComputeClosing (...)// see algorithm below.
- dilation (i, j, inAry, outAry) // Perform dilation on pixel (i, j) with structAry. // On your own!
- erosion (i, j, inAry, outAry) // Perform erosion on pixel (i, j) with structAry. // On your own!
- AryToFile (Ary, outFile) // output the img header (from input image header)
  //then output the rows and cols of Ary to outFile *excluding* the framed borders of Ary.
- prettyPrint (Ary, outFile) // Remark: use "Courier new" font and small font size to fit in the page.
  // if Ary [i, j] == 0 output ". " // a period follows by a blank
  // else output Ary [i, j] follows by a blank

****

IV. Main(...)

****

step 0: imgFile, structFile, dilateOutFile, erodeOutFile, openingOutFile, closingOutFile, prettyPrintFile ← open

step 1: numImgRows, numImgCols, imgMin, imgMax ← read from imgFile
numStructRows, numStructCols, structMin, structMax ← read from structFile
rowOrigin, colOrigin ← read from strucFile

step 2: zeroFramedAry, structAry, morphAry, tempAry ← dynamically allocate // see description in the above

step 3: zero2DAry(zeroFramedAry, numImgRows, numImgCols) // see description in the above

step 4: loadImg (imgFile, zeroFramedAry) // see description in the above
prettyPrint (zeroFramedAry, prettyPrintFile) // write a meaningful caption before prettyPrint

step 5: zero2DAry(structAry, numStructRows, numStructCols)
loadstruct (structFile, structAry) // see description in the above
prettyPrint (structAry, prettyPrintFile) // see description in the above

step 6: zero2DAry(morphAry, numImgRows, numImgCols)
ComputeDilation (zeroFramedAry, morphAry) // see algorithm below
AryToFile (morphAry, dilateOutFile) // see description in the above
prettyPrint (morphAry, prettyPrintFile) // write a meaningful caption before prettyPrint

step 7: zero2DAry(morphAry, numImgRows, numImgCols)
ComputeErosion (zeroFramedAry, morphAry) // see algorithm below
AryToFile (morphAry, erodeOutFile)
prettyPrint (morphAry, prettyPrintFile) // write a meaningful caption before prettyPrint

step 8: zero2DAry(morphAry, numImgRows, numImgCols)
ComputeOpening (zeroFramedAry, morphAry, tempAry) // see algorithm below
AryToFile (morphAry, openingOutFile)
prettyPrint (morphAry, prettyPrintFile) // write a meaningful caption before prettyPrint

step 9: zero2DAry(morphAry, numImgRows, numImgCols)
ComputeClosing (zeroFramedAry, morphAry, tempAry) // see algorithm below
AryToFile (morphAry, closingOutFile)
prettyPrint (morphAry, prettyPrintFile) // write a meaningful caption before prettyPrint

step 10: close all files

```
*******************************
V. ComputeDilation (inAry, outAry)
*******************************
        // process dilation on each pixel in the entire zeroFramedAry
step 1: i ← rowFrameSize
step 2: j ← colFrameSize
step 3: if inAry [i,j] > 0
          dilation (i, j, inAry, outAry)
step 4: j++
step 5: repeat step 3 to step 4 while j < (numImgCols + colFrameSize)
step 6: i++
step 7: repeat step 2 to step 6 while i < (numImgRows + rowFrameSize)

*******************************
VI. ComputeErosion (inAry, outAry)
*******************************
        // process dilation on each pixel in the entire zeroFramedAry
step 1: i ← rowFrameSize
step 2: j ← colFrameSize
step 3: if inAry[i,j] > 0
          erosion (i, j, inAry, outAry)
step 4: j++
step 5: repeat step 3 to step 4 while j < (numImgCols + colFrameSize)
step 6: i++
step 7: repeat step 2 to step 6 while i < (numImgRows + rowFrameSize)

*******************************
V. ComputeClosing (zeroFramedAry, morphAry, tempAry)
*******************************
step 1: ComputeDilation (zeroFramedAry, tempAry)

step 2: ComputeErosion (tempAry, morphAry)

*******************************
V. ComputeOpening (zeroFramedAry, morphAry, tempAry)
*******************************
step 1: Compute Erosion (zeroFramedAry, tempAry)

step 2: ComputeDilation (tempAry, morphAry)
```