Project 5 (in Java): Given a binary image, the task is to produce a loss-less compression of the input image via the skeleton of 8-connectness distance transform. (Read all the lecture notes on this topic posted in Google classroom.)

Summary of what your program will do:

1) Allocate two 2D arrays with extra 2 rows and extra 2 cols. One for input, called ZeroFramedAry, and one for skeleton, skeletonAry; zero frame both arrays; load input into inside of the frame of ZeroFramedAry.

2) Performs the 1st-pass of the 8-connectness distance transform for all pixels inside the frame of ZeroFramedAry.

3) reformatPrettyPrint of the result of the Pass-1 to outFile1 with proper captions.

4) Performs the 2nd-pass of the 8-connectness distance transform on the result of 1st pass (inside of the frame)

5) reformatPrettyPrint of the result of the Pass-2 to outFile1 with proper captions.

6) Performs local maxima operation on the result of 2nd-pass.

7) reformatPrettyPrint the local maxima to outFile1 with proper captions.

8a) write the header to skeleton file
8b) Produce skeleton (compressed file): for each skeleton (i, j) > 0 (i.e., local maxima),
    write a triplet i j skeleton (i,j) to *skeleton* file,
    one triplet per text-line
    // skeleton file is the compressed (skeleton) file.

9) The name of the compressed file is to be created during the run time of your program, using the original file name with an extension "_skeleton." For example, if the name of the input file is "image1", then the name of the compressed file should be "image1_skeleton".

10) close the compressed file (image1_skeleton)

// To make sure your program works correctly; you are going to do a de-compression on the compressed file as follows.

11) re-open the compressed file (image1_skeleton).

12) re-set ZeroFramedAry to zero

13) Load triplets from compressed file to ZeroFramedAry, i.e., for
    each triplet (i, j, dist), ZeroFramedAry(i, j) ← dist

14) Perform 1st-pass expansion on the ZeroFramedAry
    // algorithm given below

15) reformatPrettyPrint of the result of 1st-pass expansion to outFile2 with captions.

16) Perform 2nd pass expansion on the result of 1st expansion
    // algorithm given below

17) reformatPrettyPrint of the result of 2nd-pass expansion to outFile2 with caption.

// If your program work correctly, the result of 2nd-pass expansion should be
// identical to the result of the 2nd pass of distance transform.

18) Produce decompressed file:

```
        a) Write the original image header to the decompressed file
        b) Threshold ZeroFramedAry with threshold value == 1 begins at (1,1)
                and ends at (?,?)
          i.e., if ZeroFramedAry (i, j) >= 1
                                output 1 and a blank space to de-compressed file.
                        else
                                output 0 and a blank space to de-compressed file.
```

19) The name of the decompressed file is to be created during the run time of your program, using the name of the input file with an extension "_decompressed." For example, if the name of the input file is "image1", then the name of the compressed file should be "image1_decompressed". (This can be done simply using string concatenation.)

20) Closed the de-compressed file.

// after this step your directory should have these three files: image1, image1_skeleton, and image1_decompressed.

21) If your program works correctly, image1_decompressed should be identical to image1.

22) run your program twice: with image1 and image2

Include in your hard copies:

- cover page
- source code
- Run on image1
    - Print the input file
    - Print outFile1
    - Print outFile2
    - Print skeleton file
    - Print decompressed file

- Run on image2
    - Print the input file
    - Print outFile1
    - Print outFile2
    - Print skeleton file
    - Print decompressed file

****

Language: Java

****

Points: 12 pts

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

12/12 on time: 3/30/2021 Tuesday before midnight

+1 early submission: 3/27/2021 Saturday before midnight

-1 for 1 day late: 3/31/2021 Wednesday Thursday before midnight

-2 for 2 days late: 4/1/2021 Thursday before midnight

-12/12 : after 4/1/2021 Thursday after midnight

-6/12: does not pass compilation

0/12: program produces no output

0/12: did not submit hard copy.

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

****

I. Input (args[0]): a binary image

****

II. Outputs:

- OutFile1 (args[1]): for
    - reformatPrettyPrint of t the results of 1st pass 8-connectness distance transform
    - reformatPrettyPrint of the results of 2nd pass 8-connectness distance transform
    - reformatPrettyPrint of the local maxima skeleton

- OutFile2 (args[2]): for
    - reformatPrettyPrint of the results of 1st pass expansion
    - reformatPrettyPrint of the results of 2nd pass expansion
    - skeleton file (generated at run-time) for store the compressed file
      using the following format:
      Example:

```
20 20 0 7 // the header of the distance transform image.
4 7 2           // the skeleton pixel at (4, 7) with distance of 2
6 7 3           // the skeleton pixel at (6, 7) with distance of 3
:
:
```

- DeCompressed file (generated at run-time), an image file where
  the first text-line is the image header, follows by rows and cols of pixel values.

****

III. Data structure:

****

- An ImageProcessing class
    - numRows (int)
    - numCols (int)
    - minVal (int)
    - maxVal (int)
    - newMinVal (int)
    - newMinVal (int)

    - zeroFramedAry (int **) a 2D array, need to dynamically allocate
      of size numRows + 2 by numCols + 2.

    - skeletonAry (int **) a 2D array, need to dynamically allocate
      of size numRows + 2 by numCols + 2.

    - methods:

    - setZero (Ary) // set 2D Ary to zero. You should know how to do this.

    - loadImage (...)
      // Read from the given File onto inside frame of zeroFramedAry
      // You should know how to do this.

    - Compute8Distance (...) // See algorithm below

    - fistPass8Distance (Ary) // algorithm is given in lecture notes

    - secondPass_8Distance (zeroFramedAry) // algorithm is given in lecture notes
      // Note** In second pass, you need
      // to keep track the newMinVal and newMaxVal
      // You should know how to do this

```
- isLocalMaxima (zeroFramedAry, i, j) // algorithm is given in lecture notes
- computeLocalMaxima (zeroFramedAry, skeletonAry) // algorithm is given in lecture notes

- extractLocalMaxima(...)
        // for each skeletonAry[i,j] > 0 write the triplet to
        // skeletonFile. For easy programming, i and j do not need to
        // subtract by 1 when output the triplets to skeletonFile.

- skeletonExtraction (...) // See algorithm below

- skeletonExpansion(...) // See algorithm below

- firstPassExpension (...)// algorithm is given in lecture note.

- secondPassExpension (...)// algorithm is given in lecture note.

- ary2File(...)
                // do a threshold on zeroFramedAry
                // with the threshold value at 1, begins at (1,1)
                // and ends at (?,?)
                 i.e., if zeroFramedAry (i, j) >= 1
                        output 1 and a blank space to decompressed file.
                else
                        output 0 and a blank space to decompressed file.

- reformatPrettyPrint (…) // reuse codes from your previous project.
```

****

III. main (…)

****

```
step 0: inFile ← open input file
    numRows, numCols, minVal, maxVal ← read from inFile
    dynamically allocate zeroFramedAry with extra 2 rows and 2 cols
    dynamically allocate skeletonAry with extra 2 rows and 2 cols
    open outFile_1, outFile_2

Step 1: skeletonFileName ← args[0] + "_skeleton.txt"
Step 2: skeletonFile ← open ( skeletonFileName )

Step 3: decompressedFileName ← args[0] + "_decompressed.txt"
Step 4: decompressFile ← open (decompressedFileName)

step 5: setZero (zeroFramedAry)
     setZero (skeletonAry)

Step 6: loadImage (inFile, zeroFramedAry) // begins at zeroFramedAry (1,1)

Step 7: compute8Distance (zeroFramedAry, outFile1)  // Perform distance transform

Step 8: skeletonExtraction (zeroFramedAry, skeletonAry, skeletonFile, outFile1)
        // perform lossless compression

Step 9: skeletonExpansion (zeroFramedAry, skeletonFile, outFile2)
        // perform decompression

step 10: Output numRows, numCols, newMinVal, newMaxVal to decompressFile

Step 11: ary2File (zeroFramedAry, decompressFile)

Step 12: close all files
```

```
******************************
IV. Compute8Distance (zeroFramedAry, outFile1)
******************************
step 1: fistPass_8Distance (zeroFramedAry) /

step 2: reformatPrettyPrint (zeroFramedAry, outFile1)
                    // with proper caption i.e., 1st pass distance transform

step 3: secondPass8Distance (zeroFramedAry) // begins at zeroFramedAry(?,?)

Step 4: reformatPrettyPrint (zeroFramedAry, outFile1)
                    // with proper caption i.e., 2nd pass distance transform

******************************
V. skeletonExtraction (zeroFramedAry, skeletonAry, skeletonFile, outFile1)
******************************
step 1: computeLocalMaxima (zeroFramedAry, skeletonAry)

Step 2: reformatPrettyPrint (skeletonAry, outFile1)
                    // with proper caption i.e., Local maxima

step 3: extractLocalMaxima (skeletonAry, skeletonFile)

Step 4: close skeletonFile

******************************
VI. skeletonExpansion (zeroFramedAry, skeletonFile, outFile2)
******************************
Step 1: re-open skeletonFile

Step 2: setZero (zeroFramedAry)

step 3: load (skeletonFile, zeroFramedAry)

step 4:  firstPassExpension (zeroFramedAry)

step 5: reformatPrettyPrint (zeroFramedAry, outFile2)
                    // with proper caption i.e., 1st pass Expansion

step 6: secondPassExpension (zeroFramedAry) // begins at ZeroFramedAry(?,?)
                    // During the 2nd pass, you need to track the newMinVal and newMaxVal

Step 7: reformatPrettyPrint (zeroFramedAry, outFile2)
                    // with proper caption i.e., 2nd pass Expansion
```