Project 5 (Java): You are to implement both 4-connected and 8-connected component algorithms in this project. Your program let the user to choose which connectness (4-CC or 8-CC) to run the program from console. Both algorithms consist of the following stages except which neighbors are to be checked:

1) Pass-1: (As taught in class for 8-connectness and given in lecture notes)
   4-connected: check the neighbor right above, and left.
   8-connected: check upper 3 neighbors and the left neighbor;
2) Pass-2: (As taught in class for 8-connectness and given in lecture notes)
   4-connected: check itself, the neighbors below and the right.
   8-connected: check itself, the lower 3 and the right neighbors;
3) Manage EQ table: (Algorithm is taught in class and it is in lecture note)
4) Pass-3 (for both 4- and 8- connectness): processing the entire imgAry L to R & T to B, begins at (1,1)
   Pass-3 accomplishes the followings:
   i) re-labelling: It Uses the EQAry to relabel the connected components (CC)on the result of pass-2;
      i.e., p(i,j) ← EQAry[p(i,j)]
   ii) During the re-labelling process, it also computes all properties for each connected component (see the list of properties below), and keep track newMin and newMax for the image header of the re-labelled image.

*** You will be given three (3) data files: data1, data2 and data3. data1 is a very small image.
What do you need to do as follows:
a) Implement your program based on the specs below.
b) Test and debug your program using data1 for 4-connected until it produces the correct result.
c) Test and debug your program using data1 for 8-connected until it produces the correct result.

d) Then, for data2 and data3, run your program twice; first using 4 and then using 8.

Your hard copies include:
- Cover page
- Source code
- RFprettyPrintFile for 4-connectness for data2
- labelFile for 4-connectness for data2
- propertyFile for 4-connectness for data2
- RFprettyPrintFile for 8-connectness for data2
- labelFile for 8-connectness for data2
- propertyFile for 8-connectness for data2

- RFprettyPrintFile for 4-connectness for data3
- labelFile for 4-connectness for data3
- propertyFile for 4-connectness for data3
- RFprettyPrintFile for 8-connectness for data3
- labelFile for 8-connectness for data3
- propertyFile for 8-connectness for data3

****

Language: Java
Project points:12 pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):
   -0 (12/12 pts): on time, 10/17/2021 Sunday before midnight
   +1 (13/12 pts): early submission, 10/13/2021, Wednesday before midnight
   -1 (11/12 pts): 1 day late, 10/18/2021 Monday before midnight
   -2 (10/12 pts): 2 days late, 10/19/2021 Tuesday before midnight
   (-12/12 pts): non submission, 10/19/2021 Tuesday 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.

****

I. Inputs (args[0]):

a) A binary image.

b) whichConnectness: from console

II. Outputs:

a) RFprettyPrintFile (args[1]): (include in your hard copy) for the followings:

** a proper caption means the caption should say what the printing is.

- reformatPrettyPrint of the result of the Pass-1 with proper captions
- print newLabel and the EQAry after Pass-1, with proper captions
- reformatPrettyPrint of the result of the Pass-2 with proper captions
- print newLabel and the EQAry after Pass-2, with proper captions
- Print the EQAry after manage the EQAry, with proper caption
- reformatPrettyPrint of the result of the Pass-3 with proper captions
- reformatPrettyPrint of the result bounding boxes drawing.

b) labelFile (args[2]): to store the result of Pass-3 -- the labelled image file with image header, numRows numCols newMin NewMax. ** This file to be used in future processing.

c) propertyFile (args[3]): ** (include in your hard copy)

To store the connected component properties.

The format is to be as below:

- 1st text-line, the header of the input image,
- 2nd text-line is the total number of connected components.
- from 3rd text, use four (4) text-lines per each connected component:
- label
- number of pixels
- upperLftR upperLftC //the r c coordinated of the upper left corner
- lowerRgtR lowerRgtC //the r c coordinated of lower right corner

For an example:

```
45  40  0   9  // image header
9                  // there are a total of 9 CCs in the image
1                  // CC label 1
187         // 187 pixels in CC label 1
4     9    // upper left corner of the bounding box at row 4 column 9
35  39  // lower right corner of the bounding box at row 35 column 39
:                  :
```

** This file to be used in future processing.

****

III. Data structure:

****

- A CClabel class
  - (int) numRows
  - (int) numCols
  - (int) minVal
  - (int) maxVal
  - (int) newMin
  - (int) newMax
  - (int) newLabel // initialize to 0
  - (int) trueNumCC // the true number of connected components in the image
    // It will be determined in manageEQAry method.
  - (int) zeroFramedAry[][] // a 2D array, need to dynamically allocate
    //at run time of size numRows + 2 by numCols + 2.
  - (int) NonZeroNeighborAry [5] // 5 is the max number of neighbors you have to check.
    // For easy programming, you may consider using this 1-D array
    // to store pixel(i, j)'s non-zero neighbors during pass 1 and pass2.

- (int) EQAry [] // an 1-D array, of size (numRows * numCols) / 4
  // dynamically allocate at run time, and initialize to its index, i.e., EQAry[i] = i.
- Property (1D struct or class)
  - (int) label // The component label
  - (int) numpixels // total number of pixels in the cc.
  - (int) minR // with respect to the input image.
  - (int) minC // with respect to the input image.
  - (int) maxR // with respect to the input image.
  - (int) maxC // with respect to the input image.
  // In the Cartesian coordinate system, any rectangular box can be represented by two points: upper-left corner and the lower-right of the box. Here, the two points:(minR minC) and(maxR maxC) represents the smallest rectangular box that the cc can fit in the box; object pixels can be on the border of the box.

- (Property) CCproperty []
  // A struct 1D array for storing all components' properties.
  // The size of array is the actual number of cc after manageEQAry

- methods:
  - constructor(...) // need to dynamically allocate all arrays; and assign values to numRows,, etc.
  - zero2D (...) // ** Initialized a 2-D array to zero. You must implement this method, don't count on Java.
  - minus1D (...) // ** Initialized a 1-D array to -1.
  - loadImage (...)
    // read from input file and write to zeroFramedAry begin at(1,1)
  - imgReformat (zeroFramedAry, RFprettyPrintFile) // Print zeroFramedAry to RFprettyPrintFile
  - connect8Pass1 (...) // On your own, as taught in class and algorithm is in lecture note
  - connect8Pass2 (...) // On your own, as taught in class and algorithm is in lecture note
  - connect4Pass1 (...) // On your own, as taught in class and in lecture note
  - connect4Pass2 (...) // On your own, as taught in class and in lecture note
  - connectPass3 (...) // On your own. There is no differences between 4-connectness and 8-connectness.
  - drawBoxes (...) // Draw the bounding boxes on all connected components in zeroFramedAry.
    // See algorithm below
  - updateEQ (...) // Update EQAry for all non-zero neighbors to minLabel, it will be easier to use
    //NonZeroNeighborAry to store all non-zero neighbors.
  - (int) manageEQAry (...) // The algorithm was taught in class and in lecture note.
    // The method returns the true number of CCs in the labelled image.
  - printCCproperty (...) // Prints the component properties to propertyFile using the format given in the above.
    // On your own.
  - printEQAry (...) // Print EQAry with index up to newLabel, not beyond. On your own
  - printImg (...) // Output image header and zeroFramedAry (inside of framing) to labelFile
    // on your own.

****
IV. main(...)
****
step 0: inFile ← open the input file
  RFprettyPrintFile , labelFile, propertyFile ← open from args[]
  numRows, numCols, minVal, maxVal ← read from inFile
  dynamically allocate zeroFramedAry.
  newLabel ← 0
step 1: zero2D (zeroFramedAry)
step 2: loadImage (inFile, zeroFramedAry)
step 3: Connectness ← ask user from console
step 4: if connectness == 4
    connect4Pass1 (…)
    imgReformat (zeroFramedAry, RFprettyPrintFile)
    printEQAry (newLabel, RFprettyPrintFile)

```
                    // print the EQAry up to newLable with proper caption
            Connect4Pass2 (…)
            imgReformat (zeroFramedAry, RFprettyPrintFile)
            printEQAry (newLabel, RFprettyPrintFile)
                    // print the EQAry up to newLabel with proper caption
step 5: if connectness == 8
            connect8Pass1 (… )
            imgReformat (zeroFramedAry, RFprettyPrintFile)
            printEQAry (newLabel, RFprettyPrintFile)
                    // print the EQAry up to newLabel with proper caption
            Connect8Pass2 (…)
            imgReformat (zeroFramedAry, RFprettyPrintFile)
            printEQAry (newLabel, RFprettyPrintFile)
                    // print the EQAry up to newLabel with proper caption
step 6: trueNumCC ← manageEQAry (EQAry, newLabel)
            printEQAry (newLabel, RFprettyPrintFile)
            // print the EQAry up to newLabel with proper caption
step 7: connectPass3 (...)
step 8: imgReformat (zeroFramedAry, RFprettyPrintFile)
step 9: printEQAry (newLable, RFprettyPrintFile)
            // print the EQAry up to newLabel with proper caption
step 10: output numRows, numCols, newMin, newMax to labelFile
step 11: printImg (labelFile) // Output the result of pass3 inside of zeroFramedAry
step 12: printCCproperty (propertyFile) // print cc properties to propertyFile
step 13: drawBoxes(zeroFramedAry, CCproperty) // draw on zeroFramed image.
step 14: imgReformat (zeroFramedAry, RFprettyPrintFile)
step 15: print trueNumCC to RFprettyPrintFile with proper caption
step 16: close all files

******************************
VI. drawBoxes (zeroFramedAry, CCproperty)
******************************
// This method may contain bugs, report bugs to Dr. Phillips

step 1: index ← 1

step 2:   minRow ← CCproperty[index]'s minR  + 1
          minCol ← CCproperty[index]'s minC + 1
          maxRow ← CCproperty[index]'s maxR + 1
          maxCol ← CCproperty[index]'s maxC + 1
          label ← CCproperty[index]'s label

step 3: Assign all pixels on minRow from minCol to maxCol ← label
          Assign all pixels on maxRow from minCol to maxCol ← label
          Assign all pixels on minCol from minRow to maxRow ← label
          Assign all pixels on maxCol from minRow to maxRow ← label

step 4: index++
step 5: repeat step 2 to step 4 while index <= the actual number of cc
```