Project 4 (C++): You are to implement both 4-connected and 8-connected component algorithms in this project. Your program let the user to choose which algorithm (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:

****

Language: C++
Project points:12 pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):

3/21/2021 Sunday before midnight
+1 early submission: 3/18/2021 Thursday before midnight
-1 for 1 day late: 3/22/2021 Monday before midnight
-2 for 2 days late: 3/23/2021 Tuesday before midnight
-12/12 : after 3/23/2021 Tuesday 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.

*** Run your program twice; first using 8 and then using 4.

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

****

I. Inputs: There are two inputs
a) inFile (argv[1]): A binary image.
b) connectness (argv[2])

****

II. Outputs: There are 3 output files.

a) RFprettyPrintFile (argv[3]): (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 (argv[4]): ** (include in your hard copy)
to store the result of Pass-3 -- the labelled image file with image header, numRows numCols newMin NewMax.
** This file will be used in future processing.

c) propertyFile (argv[5]): ** (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 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 will 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.
- 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
  - (int) minC
  - (int) maxR
  - (int) maxC
  // In the Cartesian coordinate system, any rectangular box can be //represented by two points: upper-left corner and the lower-right //corner of the box. Here, the two points:(minR minC) and(maxR maxC) represents the smallest rectangular box that the cc can fit inside 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
    // as in your previous project.
  - 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 (...) // There is no differences between 4-connectness and
    // 8-connectness. On your own, as taught in class and in lecture note
  - 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 (...) // on your own
    // The algorithm was taught and given in class and in lecture note
    // The method returns the true number of CCs in
    // the labelled image.
  - printCCproperty (...) // print the component properties to propertyFile
    // using the format given in the above.
    // you should know how to do this.
  - 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 (argv[1])
RFprettyPrintFile, labelFile, propertyFile ← open from args[]
numRows, numCols, minVal, maxVal ← read from inFile
dynamically allocate zeroFramedAry.
Connectness ← from argv[2]
newLabel ← 0

step 1: zero2D (zeroFramedAry)
step 2: loadImage (inFile, zeroFramedAry)
step 3: 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 4: 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 newLable with proper caption

step 5: trueNumCC ← manageEQAry (EQAry, newLabel)
printEQAry (newLabel, RFprettyPrintFile)
// print the EQAry up to newLabel with proper caption

step 6: connectPass3 (...)
step 7: imgReformat (zeroFramedAry, RFprettyPrintFile)
step 8: printEQAry (newLabel, RFprettyPrintFile)
// print the EQAry up to newLabel with proper caption
step 9: output numRows, numCols, newMin, newMax to labelFile
step 10: printImg (labelFile) // Output the result of pass3 from zeroFramedAry to //labelFile, begins at zeroFramedAry[1, 1] and ending at ??

step 12: printCCproperty (propertyFile) // print cc properties to propertyFile
step 13: drawBoxes (zeroFramedAry, CCproperty)
step 14: imgReformat (zeroFramedAry, RFprettyPrintFile)
step 15: print trueNumCC to RFprettyPrintFile with proper caption
step 16: close all files

****
VI. drawBoxes (zeroFramedAry, CCproperty)
****

step 1: index ← 1

step 2: minRow ← CCproperty[index]'s minR // need to add 1
minCol ← CCproperty[index]'s minC // need to add 1
maxRow ← CCproperty[index]'s maxR // need to add 1
maxCol ← CCproperty[index]'s maxC // need to add 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 is within the number of cc.