Project 5: (a) You are to implement the 8-connected component algorithm taught in class. (b) After the 3rd pass, you are to draw bounding boxes of each of the connected components (over laying). Pixels on the bounding box should have the same label as the label of the connected component.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Language: C++

Due date: soft copy: 3/29/2019 Friday before Midnight

Early submission +1 deadline: 3/27/2018 Wednesday before Midnight

-1 pt due: 3/30/2019 Saturday before midnight

After 3/30/2019 -12 pts for all students who did not submit soft copy

Due Date: Hard copy: 4/2/2019 Tuesday in class,

-1 pt for late hard copy submission after Thursday 4/3/2019 (place under door: A218).

All projects without hard copy after 4/3/2019 will receive 0 pts even you have submit soft copy on time and even if it works.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. Input (argv[1]): A binary image.

II. Outputs:

- outFile1 (argv[2]): \*\* (include in your hard copy)

- Pretty print the result of the Pass-1 \*and\*

the EQAry with proper captions

- Pretty print the result of the Pass-2 \*and\*

the EQAry with proper captions

- Print EQAry after manage the EQAry

- Pretty print the result of the Pass-3 \*and\*

the EQAry with proper caption

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

- outFile2 (argv[3]): (Do NOT include in your hard copy)

the labelled image file from the result of Pass-3

with header information for future processing.

- outFile3 (argv[4]): \*\* (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

- minRow, minCol //the r c coordinated of the upper left corner

- maxRow, maxCole //the r c coordinated of lower right corner

For an example:

45 40 0 8 // image header

8 // there are a total of 8 CC 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

:

:

- outFile4 (argv[5]): \*\* (include in your hard copy), pretty print the bounding boxed connected components.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. Data structure:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- newMin (int)

- newMax (int)

- newLabel (int) // initialize to 0

- zeroFramedAry (int \*\*) // a 2D array, need to dynamically allocate

//at run time of size numRows + 2 by numCols + 2.

- numNb (int) // number of neighbors to be looked at.

// 8-connected is set to 4, 4-connected is set to 2

- NeighborAry[numNb](int) // to store pixel(i, j)’s neighbors

- EQAry (int \*)

// an 1-D array, need to dynamically allocate at run time

// of size (numRows \* numCols) / 4

// and initialize to EQAry[i] = i.

- Property (use 1D struct or class)

- label

- numpixels

- minRow

- minCol

- maxRow

- maxCol

- CCproperty (\*Property) // a array of 1D property struct/class

- methods:

- constructor(s) // need to dynamically allocate all arrays;

and assign values to numRows,..., etc.

- zeroFramed // zero framing the image as taught in class

- loadImage // read from input file and write to zeroFramedAry begin at(1,1)

- loadNeighbors // load the respective neighbors of given pixel(i,j)

- 8ConnectCC\_Pass1 // Process zeroFramedAry begins at (1,1) and end at ??

- 8ConnectCC\_Pass2 // Process zeroFramedAry begins at ?? and end at (1,1)

- 8ConnectCC\_Pass3 // In the pass3, you will use the EQAry

// to relabel the components;

// keep track the newMin newMax for the label image header

// as well as compute the property of each c.c.

// and store the cc label i’s properties to CCproperty[i]

// please note that the bounding box is computed from

// zeroFramedAry, therefore, the actual

// (minRow, minCol, maxRow, maxCol)

// need to be subtract by 1 for these four numbers.

- drawBoxes (...) // Draw the bounding boxes for all connected components.

// See algorithm below

- updateEQAry (...) // on your own.

- manageEQAry (...) // algorithm was given in class.

- printCCproperty // print the component properties.

- prettyPrint (outFile) // prettyPrint zeroFramedAry

// to outFile zeroFramedAry

//begins at (1,1) and end at ??

- printEQAry // Print EQAry with index up to newLable, not beyond.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. main(...)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: inFile 🡨 open the input file

numRows, numCols, minVal, maxVal 🡨 read the image header

dynamically allocate zeroFramedAry and initialize to zero.

outFile1, outFile1, outFile3, outFile4 🡨 open

step 1: loadImage(inFile, zeroFramedAry)

// read from input file and write to zeroFramedAry begin at(1,1)

step 2: 8ConnectCCPass1 (...) // as taught in class

prettyPrint (outFile1) // the result of pass1

printEQAry (outFile1)// with index up to newLable with proper caption

step 3: 8ConnectCCPass2 (...)// as taught in class

prettyPrint (outFile1) // the result of pass2

printEQAry (outFile1) // with index up to newLable with caption

step 4: manageEQAry (...)// algorithm was given in class.

printEQAry (outFile1)// with index up to newLable with caption

step 5: 8ConnectCCPass3 (...) // as taught in class

prettyPrint (outFile1) // the result of pass3

printEQAry (outFile1) // with index up to newLable with caption

step 6: output numRows, numCols, newMin, newMax to outFile2

step 7: Output the result of pass3 from zeroFramedAry to outFile2,

begins at (1, 1) and ending at ??

step 8: printCCproperty (...)to outFile 3

step 9: drawBoxes(zeroFramedAry, CCproperty)

step 10: output zeroFrameAry to outFile4

step 11: close all files

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V. drawBoxes(zeroFramedAry, CCproperty)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 1: index 🡨 1

step 2: minRow 🡨 CCproperty[i].minRow // need to add 1

minCol 🡨 CCproperty[i].minCol // need to add 1

maxRow 🡨 CCproperty[i].maxRow // need to add 1

maxCol 🡨 CCproperty[i].maxCol // need to add 1

label 🡨 CCproperty[i].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 – step 4 while index is within the number of cc.