Project 9.1: Implementation of the four basic Morphology Operations

You will be given two sets of data:

set1) Morphology\_Img1.txt and Morphology\_StrucElem1.txt

set2) Morphology\_Img2.txt and Morphology\_StrucElem2.txt

You are to run your program twice for each set1 and set1.

Your hard copies need to include the result for both sets.

\*\*\* Hard coded file names will be given 0 for this project!!!!!!

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

Language: C++

Due date: soft copy: 5/8/2018 Tuesday before Midnight

Early submission +1 deadline: 5/6/2018 Sunday before Midnight

Due date: hard copy: 5/10/2018 Thursday in class

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

I. Inputs: There are two input files.

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

1 Input1 (argv[1]): a txt file representing a binary image with header.

2. Input2 (argv[2]): 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:

1th 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 outputs need to be in your hard copies!)

- Console output

- Output1 (argv[3]): the result of dilation image with header, should be the same dimension as input1

- Output2 (args[4]): the result of erosion image with header, should be the same dimension as input1

- Output3 (args[5]): the result of closing image with header, should be the same dimension as input1

- Output4 (args[6]): the result of opening image with header, should be the same dimension as input1

\*\*\* Note: Please name your output file with respect to the operation. NO HARD coded file names in the program, you will receive the score of 0 for hard code file name in this project!!!

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

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

III. Data structure:

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

- numRowsImg (int)

- numColsImg (int)

- minImg (int)

- maxImg (int)

- numRowsStructElem (int)

- numColsStructElem (int)

- minStrctElem (int)

- maxStrctElem (int)

- rowOrigin (int)

- colOrigin (int)

- rowFrameSize (int)

- colFrameSize (int)

- imgAry (int \*\*) // a 2D array, to store the input image,

// needs to dynamically allocate at run time

// of size numRowsImg + rowFrameSize by numColsImg + colFrameSize.

- morphAry (int \*\*) // a 2D array, need to dynamically allocate at run time

// of size numRowsImg + rowFrameSize by numColsImg + colFrameSize.

- structElemAry (int \*\*) //a 2D array, need to dynamically allocate at run time

// of size numRowsStructElem by numColsStructElem.

- methods:

- computeFrameSize(...)

// for this project, rowFrameSize set to numRowsStructElem (half to the top and half to the bottom)

// colFrameSize set to numColsStructElem (half to the left and half to the right)

- loadImage // load imgAry from input1

- loadstruct // load structElem from input2

- zeroFrameImg // frame the input image with zero

- initMorphAry( ) // initialize morphAry to zero

- delation (i,j) // place the origin of the structuring element at pixel(i,j),

// i begins at (numRowsStructElem / 2), and j begins at (numColsStructElem / 2)

- erosion (i,j) // as above

- closing (i,j) // as above

- opening (i,j) // as above

- prettyPrint ()

- outputResult ()

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

III. Algorithm steps in main()

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

Remarks:

When you call each of the four morphological operations:

1. you need to reset the morphAry to zero,

2. scan imgAry begins at rowFrameSize and colFrameSize

3. when writing the result to the console, you must write which operation was used

4. output the result from morphAry to outfile, begins at rowFrameSize/2 and colFrameSize/2 of morphAry

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

step 0: open all files

( numRowsImg, numColsImg, minImg, maxImg ) 🡨 get from input1

( numRowsStrctElem, numColsStrctElem, minStrctElem, maxStrctElem ) 🡨 get from input2

( rowOrigin, colOrigin) 🡨 get from input2

step 1: computeFrameSize

step 2: - dynamically allocate imgAry with extra rows and extra columns

- loadImage // load input file to imgAry

- zeroFrameImg ()

- prettyPrint (imgAry) // pretty print imgAry to the \*console\* of "Input Image"

- dynamically allocate morphAry with extra rows and extra columns

step 3:

- dynamically allocate structElemAry

- loadstruct // load input2 file to structElem array

- prettyPrint (structElemAry)// pretty print to the \*console\* of "Structuring Element"

step 4: - initMorphAry( ) // initialize morphAry to zero

- call dilation // see your lecture note

- prettyPrint (morphAry) // pretty print to the \*console\* the result of dilation

- outputResult //write the delation result to Output1 (argv[3])

step 5: - initMorphAry( ) // initialize morphAry to zero

- call erosion // see your lecture note

- prettyPrint (morphAry) // pretty print to the \*console\* the result of erosion

- outputResult //write the delation result to Output2 (argv[4])

step 6: - initMorphAry( ) // initialize morphAry to zero

- call closing // By call those two morphological ops one after the other

- prettyPrint (morphAry) // pretty print to the \*console\* the result of closing

- outputResult //write the closing result to Output3 (argv[5])

step 7: - initMorphAry( ) // initialize morphAry to zero

- call opening // By call those two morphological ops one after the other

- prettyPrint (morphAry) // pretty print to the \*console\* with "Opening Result"

- outputResult //write the opening result to Output4 (argv[6])

step 8: close all files