Project 10: Morphology.

1) You are to implement the four basic Morphology Operations taught in class.

2) You will be given a binary image and a structure element using argv.

(Note: you will be given multiple sets of file to run your program)

(Extra credit option: you will be given a binary image and you are to design

your own structuring element for removing noises in the image!)

3) You will ask the user from the console which of the four morphological operations to use:

D for dilation

E for erosion

C for closing

O for opening

Q for quit

4) Repeatedly asking the user for another operation until the user type Q (quit).

5) Pretty print each intermediate result (after each operation) to file.

\*\*\* Hard coded file names (input or output) will be given 0 for this project!!!!!!

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

Language: C++

Due date: soft copy: 12/9/2018 Sunday before Midnight

Early submission +2 deadline: 12/5/2018 Wednesday before Midnight

Early submission +1 deadline: 12/7/2018 Friday before Midnight

Due date: hard copy: 12/11/2018 Thursday in class

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

I. Inputs: There are two input files.

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

a. inFile\_1 (argv[1]): a txt file representing a binary image with header.

b. inFile\_2 (argv[2]): a txt file representing a structuring element (also binary)

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 (row index and col index begin at 0)

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 dilation and 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(row index and col index begin at 0)

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. outFile\_1: (For all intermediate result!)

Console: (For debugging and see the intermediate result quickly)

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

III. Data structure:

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

- a morphology class

- 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.

- choice (char)

- 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, you should know how to do this method.

- loadstruct (…) // load structElem from input2, you should know how to do this method.

- zeroFrameImg // frame the input image with zero, you should know how to do this method.

- initMorphAry(…) // initialize morphAry to zero, you should know how to do this method.

- (char) consoleRequest (…) // first display the choices in console, then ask for his/her choice from console.

D for dilation

E for erosion

C for closing

O for opening

Q for quit

- delation (i,j) // on your own!!

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

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

// on your own!

- erosion (i,j) // as above

- closing (i,j) // as above

- opening (i,j) // as above

- consolePrettyPrint (…) // you should know how to do this method.

- outFilePrettyPrint …) // you should know how to do this method.

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

IV. main(…)

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

step 0: open all files

( numRowsImg, numColsImg, minImg, maxImg ) 🡨 get from inFile\_1

( numRowsStrctElem, numColsStrctElem, minStrctElem, maxStrctElem ) 🡨 get from inFile\_2

( rowOrigin, colOrigin) 🡨 get from inFile\_2

step 1: - computeFrameSize(…)

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

- loadImage (…)

- zeroFrameImg (…)

- consolePrettyPrint (imgAry)

- dynamically allocate morphAry with extra rows and extra columns

step 3:

- dynamically allocate structElemAry

- loadstruct (…)

- consolePrettyPrint (structElemAry)

step 4: initMorphAry(…)

choice <-- consoleRequest (…)

use case statement for the given choice

// make sure only one operation is call!!!

case ‘D’ : call dilation

case ‘E’ : call erosion

case ‘O’ : call opening

case ‘C’ : call closing

case ‘Q’ : exit the program

step 5: consolePrettyPrint (morphAry)

outFilePrettyPrint (morphAry)

step 6: repeat step 4 to step 5 until choice is Q

step 7: close all files