CV Project 3 Implementation of the four basic Morphology Operations Java

Jonathan Mathew

Project Due 02/27/23

Algorithm Steps

step 0: imgFile, elmFile, outFile1,

outFile2-->open

step 1:

numImgRows, numImgCols, imgMin, imgMax --> read from imgFile

numStructRows, numStructCols, structMin, structMax --> read from elmFile

rowOrigin, colOrigin-->read from elmFile

step 2: zeroFramedAry, structAry, morphAry, tempAry-->dynamically allocate. // see description in the above

step 3: zero2DAry(zeroFramedAry, rowSize, colSize)

step 4: loadImg (imgFile, zeroFramedAry)

step 5:

imgReformat (zeroFramedAry, outFile1) // with caption.

prettyPrint (zeroFramedAry, outFile1) // with caption.

step 6:

zero2DAry(structAry, numStructRows, numStructCols) loadstruct (structFile, structAry) prettyPrint (structAry, outFile1) // with caption.

step 7:

basicOperations (zeroFramedAry, morphAry, structAry, tempAry, outFile1)

step 8: complexOperations (zeroFramedAry, morphAry, structAry, tempAry, outFile2)

step 9: close all files.

Data1 and elm1

Output1.txt

44, 33, 0, 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0

0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 0 0 0 0

0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0

0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

44, 33, 0, 1

1

1 1 1 1

1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1

1

1 1

1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1

44, 33, 0, 1

1

1 1 1

1

entering basicOperations method

Printing result of ComputeDilation.

44, 33, 0, 1

1

1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1

Printing result of ComputeErosion.

44, 33, 0, 1

1

1 1 1

1 1 1

1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1

1

1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1

1 1

1 1

1

1

1 1

1

1 1 1

1 1 1

1 1 1

1 1

1 1

1 1 1

1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1

1

1

Printing result of ComputeOpening.

44, 33, 0, 1

1

1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1

1 1 1

1

1

1 1 1 1

1 1 1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1

1 1 1

1

Printing result of ComputeClosing.

44, 33, 0, 1

1

1 1 1 1

1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1

1

1 1

1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1

exit basicOperations method

Output2.txt

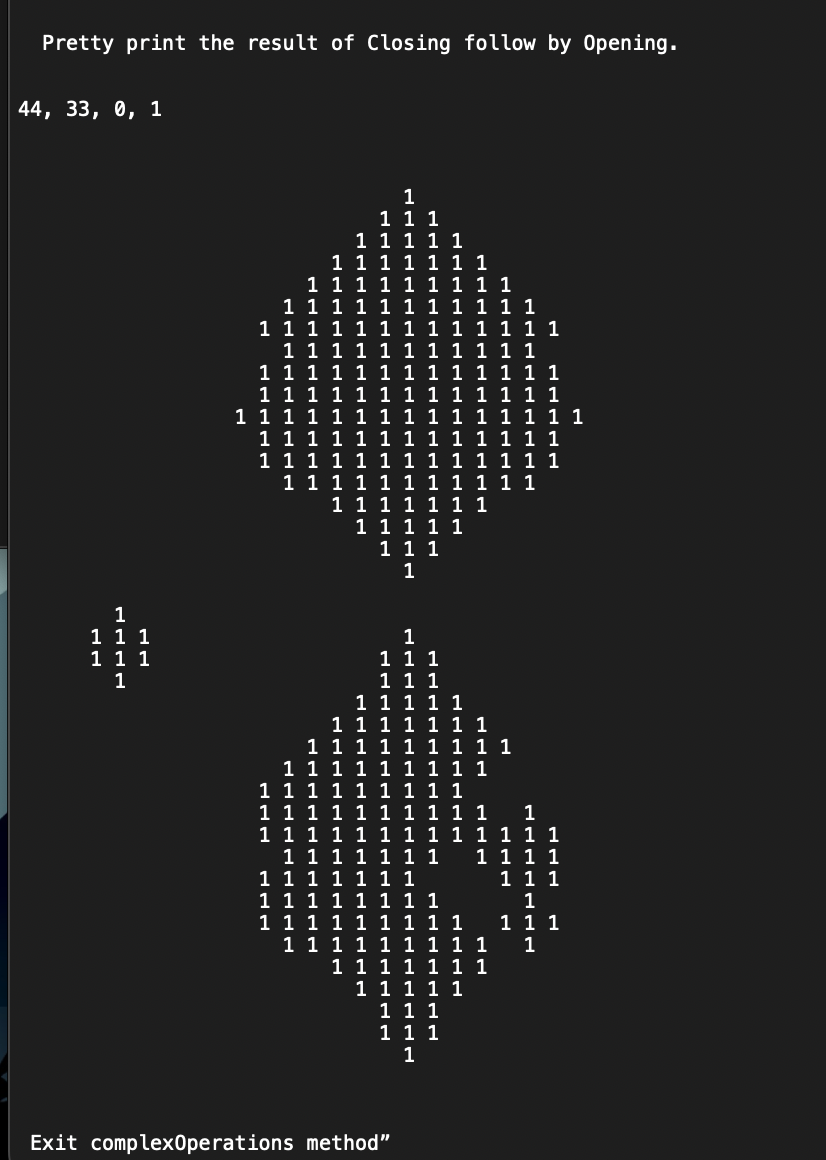
A picture containing schematic

Description automatically generated



A picture containing chart

Description automatically generated



Data2 and elm2

Output1.txt

Background pattern

Description automatically generated

A screen shot of a computer

Description automatically generated with low confidence

Text

Description automatically generated

Shape

Description automatically generated with medium confidence

Shape

Description automatically generated with medium confidence

Text

Description automatically generated

Output2.txt

Text

Description automatically generated

Text

Description automatically generated

Text

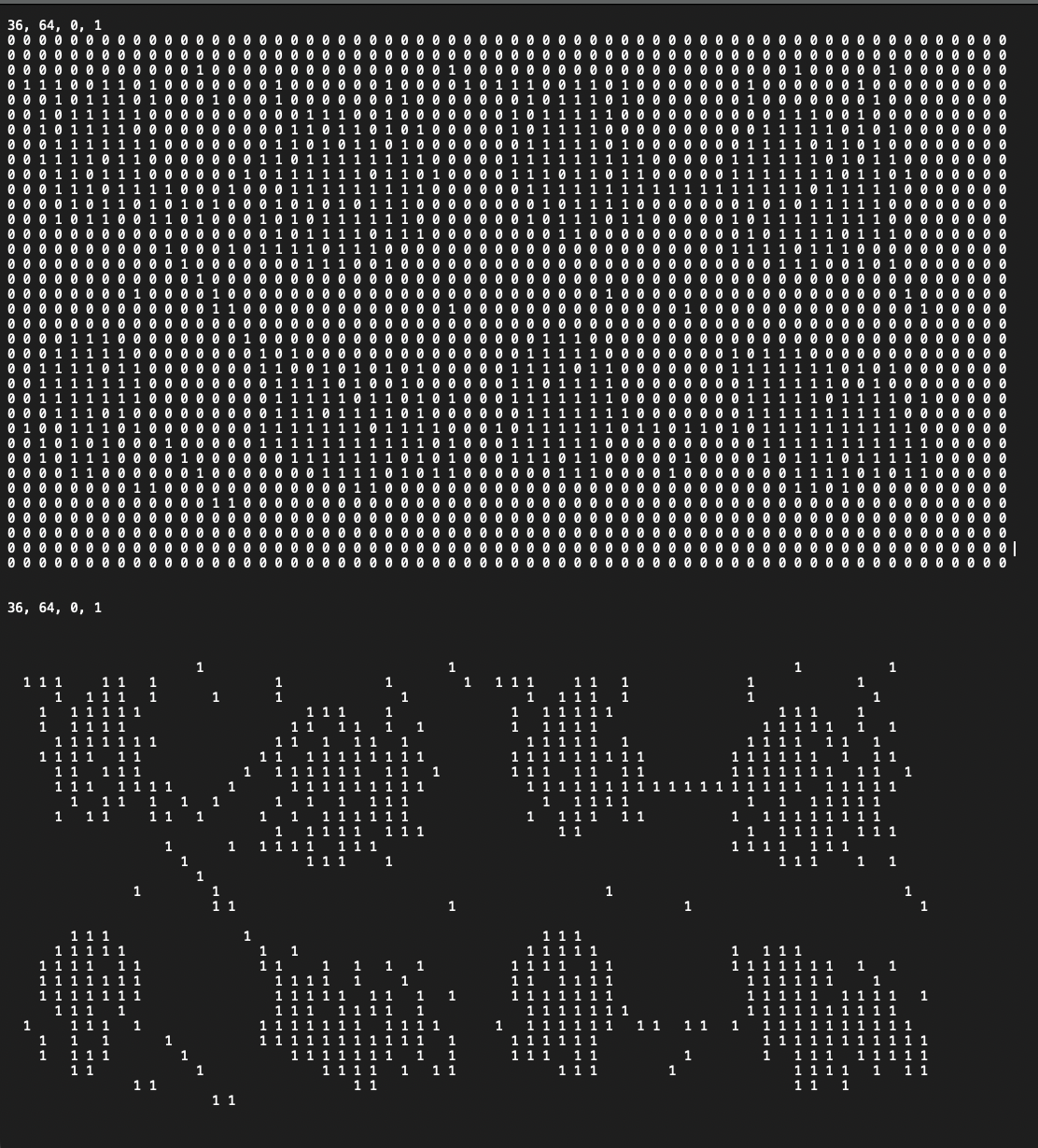
Description automatically generated

Text

Description automatically generated

Data3 and elm3

Output1.txt

Text

Description automatically generatedA picture containing text

Description automatically generated

Output2.txt

A picture containing shape

Description automatically generated

Schematic

Description automatically generated with medium confidence

Text

Description automatically generated with low confidence

A picture containing text

Description automatically generated

**Source code**

Main.java

*import* java.io.\*;  
  
*public class* Main {  
  
 *public static void* main(String[] args) *throws* IOException {  
 File img = *new* File(args[0]);  
 File elm = *new* File(args[1]);  
 File output1 = *new* File(args[2]);  
 File output2 = *new* File(args[3]);  
x  
 FileReader imgFile = *new* FileReader(img);  
 FileReader elmFile = *new* FileReader(elm);  
  
 FileWriter out1 = *new* FileWriter(output1);  
 FileWriter out2 = *new* FileWriter(output2);  
  
 *//step1* BufferedReader br = *new* BufferedReader(imgFile);  
 String line = br.readLine();  
 String[] values = line.split(" ");  
 *int* numImgRows = Integer.parseInt(values[0]),  
 numImgCols = Integer.parseInt(values[1]),  
 imgMin = Integer.parseInt(values[2]),  
 imgMax = Integer.parseInt(values[3]);  
  
 br = *new* BufferedReader(elmFile);  
 line = br.readLine();  
 values = line.split(" ");  
 *int* numStructRows = Integer.parseInt(values[0]),  
 numStructCols = Integer.parseInt(values[1]),  
 structMin = Integer.parseInt(values[2]),  
 structMax = Integer.parseInt(values[3]);  
  
 line = br.readLine();  
 values = line.split(" ");  
 *int* rowOrigin = Integer.parseInt(values[0]),  
 colOrigin = Integer.parseInt(values[1]);  
  
  
*// for (String s: values)  
// System.out.println(s);  
// System.out.println();  
  
 //step 2* Morphology proj = *new* Morphology(numImgRows, numImgCols, imgMin, imgMax, numStructRows, numStructCols, structMin, structMax, rowOrigin, colOrigin);  
  
 *//step 3  
// done in constructor  
  
 //step 4* imgFile.close();  
 imgFile = *new* FileReader(img);  
 proj.loadImg(imgFile);  
 *//step 5* proj.imgReformat(out1);  
 proj.prettyPrint(proj.zeroFramedAry, out1);  
  
 *//step 6* elmFile.close();  
 elmFile = *new* FileReader(elm);  
 proj.loadStruct(elmFile);  
 proj.prettyPrint(proj.structAry, out1);  
  
 *//step 7* proj.basicOP(out1);  
 out1.close();  
  
 *//step8* proj.complexOP(out2);  
  
 *//step 9* imgFile.close();  
 elmFile.close();  
  
 out2.close();  
  
  
 }  
}

Morphology.java

*import* java.io.BufferedReader;  
*import* java.io.FileReader;  
*import* java.io.FileWriter;  
*import* java.io.IOException;  
  
*public class* Morphology {  
  
 *int* numImgRows;  
 *int* numImgCols;  
 *int* imgMin;  
 *int* imgMax;  
 *int* numStructRows;  
 *int* numStructCols;  
 *int* structMin;  
 *int* structMax;  
 *int* rowOrigin;  
 *int* colOrigin;  
 *int* rowFrameSize;  
 *int* colFrameSize;  
 *int* extraRows;  
 *int* extraCols;  
 *int* rowSize;  
 *int* colSize;  
 *public int*[][] zeroFramedAry;  
 *public int*[][] morphAry;  
 *public int*[][] tempAry;  
 *public int*[][] structAry;  
  
  
 *public* Morphology(*int* numImgRows, *int* numImgCols, *int* imgMin, *int* imgMax, *int* numStructRows, *int* numStructCols, *int* structMin, *int* structMax, *int* rowOrigin, *int* colOrigin){  
  
 *this*.numImgRows = numImgRows;  
 *this*.numImgCols = numImgCols;  
 *this*.imgMin = imgMin;  
 *this*.imgMax = imgMax;  
 *this*.numStructRows = numStructRows;  
 *this*.numStructCols = numStructCols;  
 *this*.structMin = structMin;  
 *this*.structMax = structMax;  
 *this*.rowOrigin = rowOrigin;  
 *this*.colOrigin = colOrigin;  
  
 *this*.rowFrameSize = numStructRows /2;  
 *this*.colFrameSize = numStructCols /2;  
  
 *this*.extraRows = rowFrameSize\*2;  
 *this*.extraCols = colFrameSize\*2;  
 *this*.rowSize = numImgRows + extraRows;  
 *this*.colSize = numImgCols + extraCols;  
*//  
// System.out.println("numimg rows:" + numImgRows);  
// System.out.println("numimg col:" + numImgCols);  
// System.out.println("extraRows rows:" + extraRows);  
// System.out.println("extraCols rows:" + extraCols);  
// System.out.println("rowSize rows:" + rowSize);  
// System.out.println("colSize rows:" + colSize);  
  
 this*.zeroFramedAry = *new int*[rowSize][colSize];  
 *this*.morphAry = *new int*[rowSize][colSize];  
 *this*.tempAry = *new int*[rowSize][colSize];  
 *this*.structAry = *new int*[numStructRows][numStructCols];  
  
 zero2DAry(zeroFramedAry);  
 zero2DAry(morphAry);  
 zero2DAry(tempAry);  
 zero2DAry(structAry);  
 }  
  
 *public void* zero2DAry (*int*[][] arr){  
  
 *int* r = arr.length, c = arr[0].length;  
 *for* (*int* i=0; i<r;i++){  
 *for* (*int* j=0; j<c;j++){  
 arr[i][j]=0;  
 }  
 }  
 }  
  
 *public void* onePixelDilation(*int* i, *int* j, *int*[][] inArr, *int*[][]outArr) {  
 *int* iOffset = i - rowOrigin;  
 *int* jOffset = j - colOrigin;  
  
  
 *int* rIndex = 0;  
  
 *while* (rIndex < numStructRows) {  
 *//step 2  
 int* cIndex = 0;  
 *//step 5  
 while* (cIndex < numStructCols) {  
 *//step 3  
 if* (structAry[rIndex][cIndex] > 0) {  
*// System.out.println(iOffset + rIndex);  
// System.out.println(jOffset + cIndex);* outArr[iOffset + rIndex][jOffset + cIndex] = 1;  
 }  
  
 *//step 4* cIndex++;  
 }  
 *//step 6* rIndex++;  
 }  
  
 }  
  
 *public void* onePixelErosion(*int* i, *int* j, *int*[][] inArr, *int*[][]outArr){  
 *int* iOffset = i - rowOrigin;  
 *int* jOffset = j - colOrigin;  
 *boolean* match = *true*;  
  
 *//step1  
 int* rIndex = 0;  
  
  
 *//step 7  
 while* ((match) && (rIndex < numStructRows)) {  
 *//step 2  
 int* cIndex = 0;  
 *//step 5  
 while* ( (match) && (cIndex < numStructCols )) {  
 *//step 3  
 if* (structAry[rIndex][cIndex] > 0 &&  
 (inArr[iOffset + rIndex][jOffset + cIndex] ) <= 0) {  
 match = *false*;  
 }  
  
 *//step 4* cIndex++;  
 }  
 *//step 6* rIndex++;  
 }  
  
 *//step 8  
 if*(match){  
 outArr[i][j] = 1;  
 }*else*{  
 outArr[i][j] = 0;  
 }  
  
 }  
  
 *public void* computeDilation (*int*[][] inArr, *int*[][]outArr){  
 *//step 1  
 int* i = rowFrameSize;  
  
 *//step 7  
 while* (i<rowSize) {  
  
 *//step 2  
 int* j = colFrameSize;  
  
 *//step 5  
 while* (j < colSize) {  
  
 *//step 3  
 if* (inArr[i][j] > 0) {  
 onePixelDilation(i, j, inArr, outArr);  
 }  
  
 *//step 4* j++;  
 }  
  
 *//step 6;* i++;  
 }  
 }  
  
 *public void* computeErosion (*int*[][] inArr, *int*[][]outArr){  
 *//step1  
 int* i = rowFrameSize;  
  
 *//step 7  
 while* (i<rowSize) {  
 *//step2  
 int* j = colFrameSize;  
  
 *//step5  
  
 while* (j < colSize) {  
 *//step 3  
 if* (inArr[i][j] > 0) {  
 onePixelErosion(i, j, inArr, outArr);  
 }  
  
 *//step 4* j++;  
 }  
  
 *//step6* i++;  
 }  
 }  
  
 *public void* computeClosing(){  
 computeDilation(zeroFramedAry,tempAry);  
 computeErosion(tempAry, morphAry);  
 zero2DAry(tempAry);  
 }  
  
 *public void* computeOpening(){  
 computeErosion(zeroFramedAry, tempAry);  
 computeDilation(tempAry,morphAry);  
 zero2DAry(tempAry);  
 }  
  
 *public void* loadImg(FileReader input) *throws* IOException {  
 BufferedReader br = *new* BufferedReader(input);  
 String line = br.readLine();  
 *int* i=1;  
 *while*((line=br.readLine())!=*null*){  
 String[] c = line.split(" ");  
 *for*(*int* j=1; j<c.length; j++){  
 zeroFramedAry[i][j] = Integer.parseInt(c[j-1]);  
 }  
 i++;  
 }  
 }  
  
 *public void* loadStruct(FileReader input) *throws* IOException {  
 BufferedReader br = *new* BufferedReader(input);  
 String line = br.readLine();  
 line = br.readLine();  
 *int* i=0;  
 *while*((line=br.readLine())!=*null*){  
 String[] c = line.split(" ");  
 *for*(*int* j=0; j<c.length; j++){  
 structAry[i][j] = Integer.parseInt(c[j]);  
*// System.out.print(c[j]);* }  
*// System.out.println();* i++;  
 }  
 }  
  
 *public void* imgReformat(FileWriter output) *throws* IOException {  
 output.write("\n" + zeroFramedAry.length + ", " + zeroFramedAry[0].length + ", 0, 1\n");  
  
 *int* width = 1;  
  
 *int* r = 0;  
  
 *while* (r<zeroFramedAry.length){  
 *int* c = 0;  
 *while* (c<zeroFramedAry[0].length){  
  
 output.write(zeroFramedAry[r][c] + " ");  
 c++;  
 }  
 output.write("\n");  
 r++;  
 }  
  
 }  
  
 *public void* prettyPrint(*int*[][] arr, FileWriter output) *throws* IOException {  
  
 output.write("\n\n" + zeroFramedAry.length + ", " + zeroFramedAry[0].length + ", 0, 1\n\n");  
 *for* (*int* i = 0; i < arr.length; i++)  
 {  
 *for* (*int* j = 0; j < arr[0].length; j++)  
 {  
 *if*(arr[i][j]>0){  
 output.write(arr[i][j] + " ");  
 }*else*{  
 output.write( " ");  
 }  
 }  
 output.write("\n");  
  
 }  
 }  
  
 *public void* basicOP(FileWriter output)*throws* IOException{  
  
 *//step 0* output.write("\nentering basicOperations method \n");  
 *//step 1* zero2DAry(morphAry);  
 computeDilation(zeroFramedAry, morphAry);  
 output.write("\n Printing result of ComputeDilation. \n");  
 prettyPrint(morphAry,output);  
 *//step 2* zero2DAry(morphAry);  
 computeErosion(zeroFramedAry, morphAry);  
 output.write("\n Printing result of ComputeErosion. \n");  
 prettyPrint(morphAry,output);  
 *//step 3* zero2DAry(morphAry);  
 computeOpening();  
 output.write("\n Printing result of ComputeOpening. \n");  
 prettyPrint(morphAry,output);  
 *//step 4* zero2DAry(morphAry);  
 computeClosing();  
 output.write("\n Printing result of ComputeClosing. \n");  
 prettyPrint(morphAry,output);  
  
 *//step 5* output.write("\nexit basicOperations method\n");  
 }  
  
 *public void* complexOP(FileWriter output)*throws* IOException{  
 *//step 0* output.write("entering complexOperations method \n");  
  
 *//step 1* zero2DAry(morphAry);  
 computeOpening();  
 output.write("\n Printing result of Opening. \n");  
 prettyPrint(morphAry,output);  
 copyArr();  
  
 *//step 2* zero2DAry(morphAry);  
 computeClosing();  
 output.write("\n Pretty print the result of Opening follow by Closing. \n");  
 prettyPrint(morphAry,output);  
 copyArr();  
  
 *//step 3* zero2DAry(morphAry);  
 computeClosing();  
 output.write("\n Pretty print the result of Closing. \n");  
 prettyPrint(morphAry,output);  
 copyArr();  
  
 *//step 4* zero2DAry(morphAry);  
 computeOpening();  
 output.write("\n Pretty print the result of Closing follow by Opening. \n");  
 prettyPrint(morphAry, output);  
  
  
 output.write("\n Exit complexOperations method” \n");  
 }  
  
 *void* copyArr(){  
 *for* (*int* i=0; i<zeroFramedAry.length;i++){  
 System.arraycopy(morphAry[i], 0, zeroFramedAry[i], 0, zeroFramedAry[0].length);  
 }  
 }  
  
}