Name: Alif Rahi
Section: CSCI 381 - Computer Vision / Tues-Thurs 1:40-2:55pm
Project: 7
Due: April 27, 2023

```
*******************************
IV. Main (...)
*******************************
Step 0: labelFile, propFile ← open via args []
        numRows, numCols, minVal, maxVal ← LabelFile
        numRows, numCols, minVal, maxVal ← propFile // need this read, so you may proceed.
        numCC ← propFile
        imgAry ← dynamically allocated
        zeroFramed (imgAry)
        loadImage (labelFile, imgAry)
        reformatPrettyPrint (imgAry, deBugFile) // with caption
        CCAry ←dynamically allocated
Step 1: chainCodeFileName ← argv[1]+ "_chainCode.txt"
        BoundaryFileName ← argv[1]+ "_Boundary.txt"
        chainCodeFile ← open (chainCodeFileName)
        BoundaryFile ← open (BoundaryFileName)
        chainCodeFile ← numRows, numCols, minVal, maxVal // image header, one text line
        chainCodeFile ← numCC // one text line
Step 2: CC.label ← propFile
        CC.numPixels ← propFile
        CC.minRow ← propFile
        CC.minCol ← propFile
        CC.maxRow ← propFile
        CC.maxCol ← propFile
Step 3: clearCCAry () // zero out the old CCAry for next CC
Step 4: loadCCAry (CC.label, CCAry) // Extract the pixels with CClabel from imgAry to CCAry.
        reformatPrettyPrint (CCAry, deBugFile) // with caption
```

```
package RahiA_Project7;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class RahiA_Project7_Main {
	static class Point{
		int row=-1, col=-1;
		public Point() {}
		public Point(int row, int col) {
			this.row = row;
			this.col = col;
		}

	}
	static class CCProperty{
		int label=-1, numPixels=-1, minRow=-1, minCol=-1, maxRow=-1, maxCol=-1;
		public CCProperty() {}
	}
	static Scanner inFile, prop, chainCodeScan;
	static BufferedWriter prettyPrintFile, chainCodeFile, BoundaryFile;
	static BufferedWriter debugFile;

	static class ChainCode{
		Point point;
		CCProperty CC;
	   int numRows, numCols, minVal, maxVal, numCC;
	   int[][] imgAry, boundaryAry, CCAry;
	   Point[] coordOffset = new Point[8];
	   int[] zeroTable = {6, 0, 0, 2, 2, 4, 4, 6};
		Point startP, currentP ,nextP;
		int lastQ, nextDir, PchainDir;

	   public ChainCode(int numImgRows, int numImgCols,int imgMin,int imgMax) throws
IOException{
			this.numRows = numImgRows;
			this.numCols = numImgCols;
			this.minVal = imgMin;
			this.maxVal = imgMax;
			imgAry = new int[numImgRows+2][numImgCols+2];
			boundaryAry = new int[numImgRows+2][numImgCols+2];
			CCAry = new int[numImgRows+2][numImgCols+2];
			coordOffset[0] = new Point(0,1);
			coordOffset[1] = new Point(-1,1);
			coordOffset[2] = new Point(-1,0);
			coordOffset[3] = new Point(-1,-1);
```

```
                coordOffset[4] = new Point(0,-1);
                coordOffset[5] = new Point(1,-1);
                coordOffset[6] = new Point(1,0);
                coordOffset[7] = new Point(1,1);

                CC= new CCProperty();
                startP = new Point();
                nextP = new Point();
                currentP = new Point();

    zero2DAry(imgAry);
    zero2DAry(boundaryAry);
}



void loadImage(){
    for(int i =1; i<=numRows; i++){
        for(int j=1; j<=numCols; j++){
            imgAry[i][j] = inFile.nextInt();
        }
    }

}


void zero2DAry(int[][] ary){
    for(int i =0; i<ary.length; i++){
        for(int j=0; j<ary[i].length; j++){
            ary[i][j] = 0;
        }
    }
}

void loadCCAry(int CClabel, int[][] CCAry) {
    for(int i =0; i<CCAry.length; i++){
        for(int j=0; j<CCAry[i].length; j++){
            if(imgAry[i][j]==CClabel) {
                CCAry[i][j] = CClabel;
            }
        }
    }
}



     public void reformatPrettyPrint(BufferedWriter file, int[][] ary) throws IOException {
    for(int i =1; i<numRows+1; i++){
```

```
        for(int j=1; j<numCols+1; j++){
                                if(i<ary.length && j<ary[i].length) {
                                  if (ary[i][j] < 10){    // 2 padded spaces
                                          prettyPrintFile.write(Integer.toString(ary[i][j]) + "  ");
                        }
                        else if(ary[i][j] < 100){   // 1 padded space
                             prettyPrintFile.write(Integer.toString(ary[i][j]) + " ");
                        }
                        else{   // no spaces
                             prettyPrintFile.write(Integer.toString(ary[i][j]));
                        }
                                        }
        }
        prettyPrintFile.write("\n");

    }
         }


void prettyPrint(int[][] ary) throws IOException{
    for(int i=0; i<=numRows+1; i++){
         prettyPrintFile.write("\n");
        for(int j=0; j<=numCols+1; j++){
           if(i< ary.length && j < ary[i].length && ary[i][j] > 0 ) prettyPrintFile.write(ary[i][j]+" ");
           else prettyPrintFile.write("  ");
        }
    }

}

int mod(int n, int mod){
    int temp = n;
    while(temp < 0){
        temp += mod;
    }
    return temp%mod;
}

public int findNextP(int lastQ) {
        //Entering findNextP method

        //last zero
        int index = lastQ;
        int i, j;
        boolean found = false;
        int ans =-1;

        while(!found){
                  i = currentP.row + coordOffset[index].row;
```

```
                    j = currentP.col +  coordOffset[index].col;

                    if(imgAry[i][j] == CC.label) {

                            found = true;
                            return index;
                    }
                    else {
                    index = mod(++index, 8);
                    }

                }


                return ans;
        }


                public void getChainCode(CCProperty cc, int[][] ary, BufferedWriter chainCodeFile)
throws IOException {
                        //Entering chainCode method
                        boolean found = false;

                    for(int i=1; i<=numRows; i++){
                            if(found) break;
                    for(int j=1; j<=numCols; j++){
                       if (CCAry[i][j] == CC.label){
                            chainCodeFile.write(CC.label+" "+i+" "+j+" ");
                          startP.row = i;
                          startP.col = j;
                                    currentP.row = i;
                                    currentP.col = j;
                                    lastQ = 4;
                                    found = true;
                                    break;
                       }
                    }
                        }


                        do {
                                int nextQ = mod(++lastQ, 8);
                                PchainDir = findNextP(nextQ);

                                if(PchainDir <0) break;
                                chainCodeFile.write(PchainDir+" ");
                                nextP.row = currentP.row + coordOffset[PchainDir].row;
                                nextP.col = currentP.col +coordOffset[PchainDir].col;
                                currentP = nextP;
```

```
                        if(PchainDir == 0) {
                                lastQ = zeroTable[7];
                        }
                        else {
                                lastQ = zeroTable[PchainDir-1];
                        }
                }while(currentP.row != startP.row || currentP.col != startP.col);

        }



    public void pretty(int[][] ary) throws IOException{
        for(int i =1; i<=numRows; i++){
            if(i>1) BoundaryFile.write("\n");
            for(int j=1; j<=numCols; j++){
                if(ary[i][j] > 0){
                    if (ary[i][j] < 10){
                        BoundaryFile.write(ary[i][j]+"  ");
                    }
                    else if(ary[i][j] < 100){
                         BoundaryFile.write(ary[i][j]+ " ");
                    }
                    else{
                         BoundaryFile.write(ary[i][j]+ ary[i][j]);
                    }
                }
                else BoundaryFile.write(ary[i][j]+ "  ");

            }
        }
    }


         public void goToNext(int next) {
        switch(next){
case 0:
    currentP.row = currentP.row;
    currentP.col = currentP.col+1;
    break;
case 1:
 currentP.row = currentP.row-1;
 currentP.col = currentP.col+1;
    break;
case 2:
 currentP.row = currentP.row-1;
 currentP.col = currentP.col;
    break;
case 3:
```

```
        currentP.row = currentP.row-1;
        currentP.col = currentP.col-1;
          break;
        case 4:
        currentP.row = currentP.row;
        currentP.col = currentP.col-1;
          break;
        case 5:
        currentP.row = currentP.row+1;
          currentP.col = currentP.col-1;
          break;
        case 6:
        currentP.row = currentP.row+1;
        currentP.col = currentP.col;
          break;
        case 7:
        currentP.row = currentP.row+1;
          currentP.col = currentP.col+1;
          break;
    }

                }


        }


        private static boolean checkBoundary(int row, int col, int[][] ary) {
                return row>=0 && col>=0 && row<= ary.length && col<= ary[row].length;
        }


public static void main(String[] args) throws IOException{
        //Step 0
                File labelFile = new File(args[0]+".txt");
                File propFile = new File(args[1]+".txt");
                File outFile = new File(args[2]+".txt");
                prettyPrintFile = new BufferedWriter(new FileWriter(outFile));

                int numImgRows, numImgCols,imgMin,imgMax;
        inFile = new Scanner(labelFile);
        prop = new Scanner(propFile);
                numImgRows = inFile.nextInt();
                numImgCols = inFile.nextInt();
                imgMin = inFile.nextInt();
                imgMax = inFile.nextInt();

                ChainCode chainCode = new ChainCode(numImgRows,numImgCols,imgMin,imgMax);
```

```
                numImgRows = prop.nextInt();
                numImgCols = prop.nextInt();
                imgMin = prop.nextInt();
                imgMax = prop.nextInt();
                chainCode.numCC = prop.nextInt();

                //Repeat steps 2-5

                //Step 1
                File chainCodeFileName = new File(args[1]+ "_chainCode.txt");
                File BoundaryFileName = new File(args[1]+ "_Boundary.txt");

                 if(chainCodeFileName.createNewFile()){
               System.out.println(args[1]+ "_chainCode.txt"+" File Created");
            }else System.out.println("File "+args[1]+ "_chainCode.txt"+" already exists");

                 if(BoundaryFileName.createNewFile()){
               System.out.println(args[1]+ "_Boundary.txt"+" File Created");
            }else System.out.println("File "+args[1]+ "_Boundary.txt"+" already exists");

                 chainCodeFile = new BufferedWriter(new FileWriter(chainCodeFileName));
                 BoundaryFile =  new BufferedWriter(new FileWriter(BoundaryFileName));

                 chainCodeFile.write(numImgRows+" "+numImgCols+" "+ imgMin+"
"+imgMax+"\n"+chainCode.numCC+"\n");

                 int tmp = chainCode.numCC;

                while(tmp >  0) {
                        tmp--;

                        chainCode.loadImage();

                        chainCode.reformatPrettyPrint(prettyPrintFile, chainCode.imgAry);
                        prettyPrintFile.write("\n");


                         //Step 2

                         chainCode.CC.label = prop.nextInt();
                         chainCode.CC.numPixels = prop.nextInt();
                         chainCode.CC.minRow = prop.nextInt();
                         chainCode.CC.minCol = prop.nextInt();
                         chainCode.CC.maxRow = prop.nextInt();
                         chainCode.CC.maxCol = prop.nextInt();

                         chainCode.zero2DAry(chainCode.CCAry);
                         chainCode.loadCCAry(chainCode.CC.label, chainCode.CCAry);
                         chainCode.reformatPrettyPrint(prettyPrintFile, chainCode.CCAry);
```

```
                    chainCode.getChainCode(chainCode.CC, chainCode.CCAry, chainCodeFile);

                }

                chainCodeFile.close();


                tmp = chainCode.numCC;
                File openChainCode = new File(args[1]+"_chainCode.txt");
                chainCodeScan = new Scanner(openChainCode);

                while(chainCodeScan.hasNext()) {

                        numImgRows = chainCodeScan.nextInt();
                        numImgCols = chainCodeScan.nextInt();
                        imgMin = chainCodeScan.nextInt();
                        imgMax = chainCodeScan.nextInt();

            chainCode.numCC =  chainCodeScan.nextInt();
            chainCode.CC.label =  chainCodeScan.nextInt();
            chainCode.currentP.row = chainCodeScan.nextInt();
            chainCode.currentP.col = chainCodeScan.nextInt();

            int next;

            while(chainCodeScan.hasNext() && checkBoundary(chainCode.currentP.row,
chainCode.currentP.col, chainCode.boundaryAry)){
                chainCode.boundaryAry[chainCode.currentP.row][chainCode.currentP.col] =
chainCode.CC.label;
               next = chainCodeScan.nextInt();
               chainCode.goToNext(next);
            }
            //write to boundaryFile
            chainCode.pretty(chainCode.boundaryAry);


                }


                //important
                prettyPrintFile.close();
                prop.close();
                inFile.close();
                chainCodeFile.close();
                BoundaryFile.close();
   }

}
```

Outputs AND Debug files for **img1CC.txt**

Debug file (3 pages):

entering getChainCode method
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 5 currentP.row = 3 currentP.col =15 nextP.row = 4
nextP.col = 14
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 3 currentP.row = 4 currentP.col =14 nextP.row = 5
nextP.col = 13
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 3 currentP.row = 5 currentP.col =13 nextP.row = 6
nextP.col = 12
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 3 currentP.row = 6 currentP.col =12 nextP.row = 7
nextP.col = 11
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 3 currentP.row = 7 currentP.col =11 nextP.row = 8
nextP.col = 10
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 3 currentP.row = 8 currentP.col =10 nextP.row = 9
nextP.col = 10
entering findNextP method
leaving findNextP method
chainDir = 0

lastQ = 6 nextQ = 5 currentP.row = 9 currentP.col =10 nextP.row = 9
nextP.col = 11
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 7 currentP.row = 9 currentP.col =11 nextP.row = 9
nextP.col = 12
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 7 currentP.row = 9 currentP.col =12 nextP.row = 9
nextP.col = 13
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 7 currentP.row = 9 currentP.col =13 nextP.row = 9
nextP.col = 14
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 7 currentP.row = 9 currentP.col =14 nextP.row = 9
nextP.col = 15
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 7 currentP.row = 9 currentP.col =15 nextP.row = 10
nextP.col = 16
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 10 currentP.col =16 nextP.row = 11
nextP.col = 16
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 11 currentP.col =16 nextP.row = 12
nextP.col = 16

entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 5 currentP.row = 12 currentP.col =16 nextP.row = 13
nextP.col = 15
entering findNextP method
leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 13 currentP.col =15 nextP.row = 13
nextP.col = 14
entering findNextP method
leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 13 currentP.col =14 nextP.row = 13
nextP.col = 13
entering findNextP method
leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 13 currentP.col =13 nextP.row = 13
nextP.col = 12
entering findNextP method
leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 13 currentP.col =12 nextP.row = 13
nextP.col = 11
entering findNextP method
leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 13 currentP.col =11 nextP.row = 13
nextP.col = 10
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 3 currentP.row = 13 currentP.col =10 nextP.row = 14
nextP.col = 10
entering findNextP method
leaving findNextP method

chainDir = 7
lastQ = 4 nextQ = 5 currentP.row = 14 currentP.col =10 nextP.row = 15
nextP.col = 11
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 5 currentP.row = 15 currentP.col =11 nextP.row = 15
nextP.col = 12
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 7 currentP.row = 15 currentP.col =12 nextP.row = 16
nextP.col = 13
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 5 currentP.row = 16 currentP.col =13 nextP.row = 17
nextP.col = 14
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 5 currentP.row = 17 currentP.col =14 nextP.row = 18
nextP.col = 15
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 18 currentP.col =15 nextP.row = 19
nextP.col = 15
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 5 currentP.row = 19 currentP.col =15 nextP.row = 19
nextP.col = 16
entering findNextP method
leaving findNextP method

Output arrays:

```
. . . . . .
. . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . .

. . . . . .
. . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 1 1 . . . . . .
```

. . . . . .

ChainCode output:

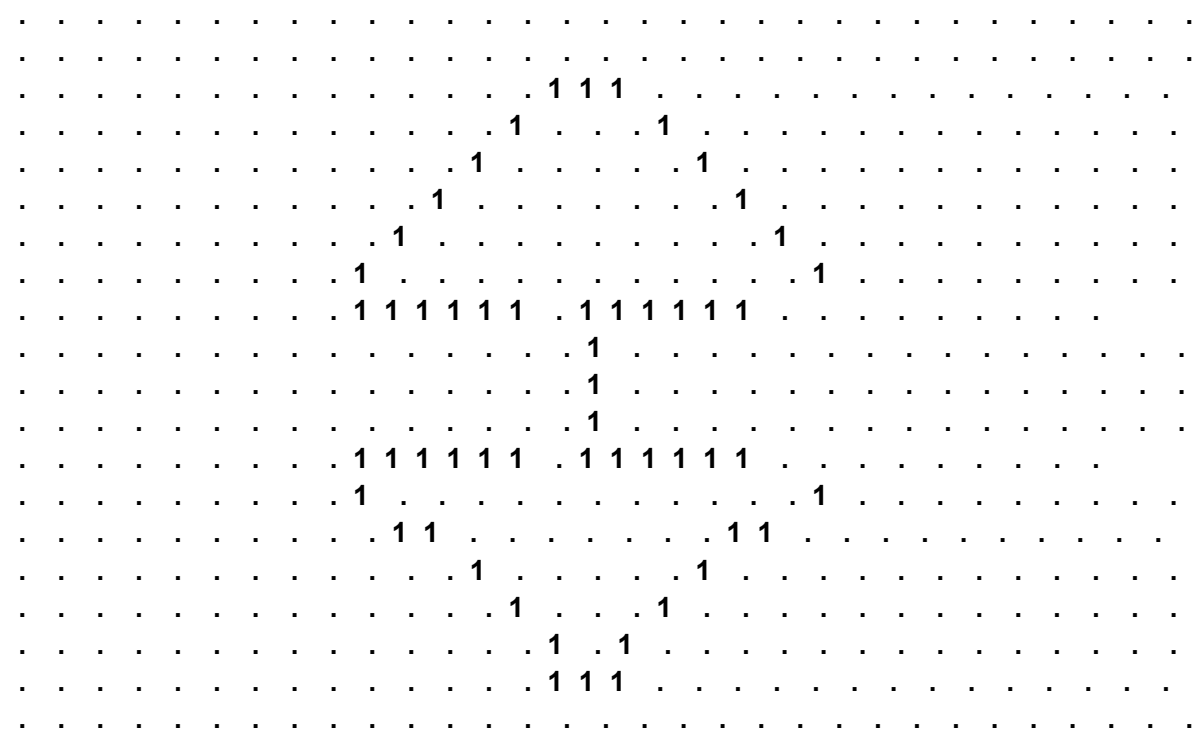
20 31 0 1

1

1 3 15 5 5 5 5 5 6 0 0 0 0 0 7 6 6 5 4 4 4 4 4 6 7 0 7 7 7 6 0 0 2

1 1 1 0 1 2 4 4 4 4 4 3 2 2 1 0 0 0 0 0 2 3 3 3 3 3 4 4

BoundaryFile output:



Outputs AND Debug files for **img2CC.txt**

Debug file (3 pages):

entering getChainCode method
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 5 currentP.row = 3 currentP.col =8 nextP.row = 4
nextP.col = 7
entering findNextP method

leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 4 currentP.col =7 nextP.row = 4
nextP.col = 6
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 3 currentP.row = 4 currentP.col =6 nextP.row = 5
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 3 currentP.row = 5 currentP.col =5 nextP.row = 6
nextP.col = 6
entering findNextP method
leaving findNextP method
chainDir = 0
lastQ = 6 nextQ = 5 currentP.row = 6 currentP.col =6 nextP.row = 6
nextP.col = 7
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 7 currentP.row = 6 currentP.col =7 nextP.row = 7
nextP.col = 8
entering findNextP method
leaving findNextP method
chainDir = 5
lastQ = 2 nextQ = 5 currentP.row = 7 currentP.col =8 nextP.row = 8
nextP.col = 7
entering findNextP method
leaving findNextP method
chainDir = 4
lastQ = 2 nextQ = 3 currentP.row = 8 currentP.col =7 nextP.row = 8
nextP.col = 6
entering findNextP method
leaving findNextP method
chainDir = 4

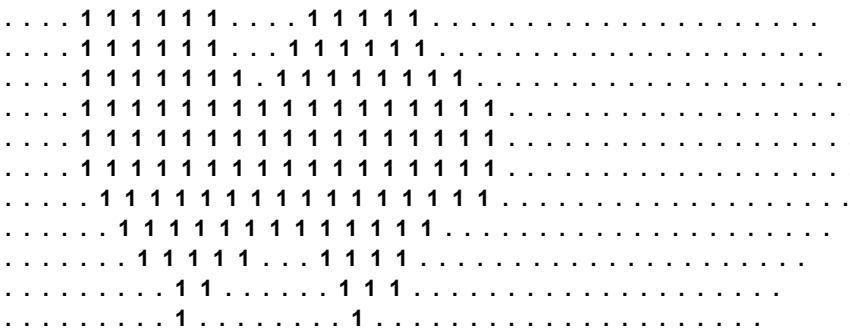
lastQ = 2 nextQ = 3 currentP.row = 8 currentP.col =6 nextP.row = 8
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 3 currentP.row = 8 currentP.col =5 nextP.row = 9
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 9 currentP.col =5 nextP.row = 10
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 10 currentP.col =5 nextP.row = 11
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 11 currentP.col =5 nextP.row = 12
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 12 currentP.col =5 nextP.row = 13
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 13 currentP.col =5 nextP.row = 14
nextP.col = 5
entering findNextP method
leaving findNextP method
chainDir = 6
lastQ = 4 nextQ = 5 currentP.row = 14 currentP.col =5 nextP.row = 15
nextP.col = 5

entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 5 currentP.row = 15 currentP.col =5 nextP.row = 16 nextP.col = 6
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 5 currentP.row = 16 currentP.col =6 nextP.row = 17 nextP.col = 7
entering findNextP method
leaving findNextP method
chainDir = 7
lastQ = 4 nextQ = 5 currentP.row = 17 currentP.col =7 nextP.row = 18

Output arrays:

```
. . . . . .
. . . . . .
. . . . . 1 . . . . . . . . . . . . . . . . . . . . 2 2 2 . . . . . . . .
. . . . . 1 1 1 1 1 . . . . . . . . . . . . . . . . . . 2 2 2 2 2 . . . . . . .
. . . . 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . 2 2 2 2 2 2 2 . . . . . .
. . . . . 1 1 1 1 1 . . . . . 1 . . . . . . . . . 2 2 2 2 2 2 2 2 2 . . . . .
. . . . . 1 1 1 . . . . . 1 . . . . . . . . 2 2 2 2 2 2 2 2 2 2 2 . . . .
. . . . 1 1 1 1 1 1 . . . . . 1 . . . . . . . . 2 2 2 2 2 2 2 2 2 2 2 . . . .
. . . . 1 1 1 1 1 1 . . . . . 1 1 1 . . . . . . . 2 2 2 2 2 2 2 2 2 2 2 . . . .
. . . . 1 1 1 1 1 1 . . . . 1 1 1 1 1 . . . . . . . 2 2 2 2 2 2 2 2 2 2 . . . . .
. . . . 1 1 1 1 1 1 . . . 1 1 1 1 1 1 . . . . . . . . 2 2 2 2 2 2 2 . . . . . .
. . . . 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . . .
. . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 3 3 3 3 3 3 3 3 3 3 3 3 3 3 . .
. . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 3 3 3 3 3 3 3 3 3 3 3 3 3 . . .
. . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . 3 3 3 3 3 3 3 3 3 3 3 . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 3 3 3 3 3 3 3 3 3 3 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . 3 3 3 3 . 3 3 3 3 3 . . . . . .
. . . . . 1 1 1 1 1 . . . 1 1 1 1 . . . . 3 3 3 3 . . 3 3 3 . . . . . .
. . . . . 1 1 . . . . . . 1 1 1 . . . 3 3 . . . . . .
. . . . . 1 . . . . . . . . 1 . . . . . 3 . . . . . .


. . . . . .
. . . . . .
. . . . . 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . .
. . . . 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . 1 . . . . . .
. . . . . 1 1 1 . . . . . 1 . . . . . .
. . . . 1 1 1 1 1 1 . . . . . 1 . . . . . .
. . . . 1 1 1 1 1 1 . . . . . 1 1 1 . . . . . .
```



ChainCode output:

20 40 0 3

3

1 3 8

5 4 5 7 0 7 5 4 4 6 6 6 6 6 6 7 7 7 0 7 6 1 1 1 0 0 7 0 7 7 1 3 2

1 0 2 2 2 3 3 2 3 3 2 2 6 6 5 5 5 5 5 3 3 2 2 2 2 2 2 4 3

2 3 30

5 5 5 5 6 6 7 7 0 0 0 0 0 0 1 1 2 2 3 3 3 3 4 4

3 13 24

7 7 5 5 6 6 7 2 1 0 2 1 7 7 0 0 1 1 1 0 1 1 4 4 4 4 4 4 4 4 4 4 4 4

4 4

BoundaryFile output: