********** Cover Page **********

Class:	CV
Name:	Frank Yournet
Project:	Project 6
Project Name:	Connected Components algorithms
Language:	Java
Due Date:	11/6/2024 before 12:00AM
Submit Date:	11/6/2024 before 12:00AM

Top Level algorithm steps

```
******************************

IV. main(...)

******************************

step 0:   inFile, prettyPrintFile, labelFile, propertyFile, logFile ← open via args []
          Connectness←args [1]

          numRows, numCols, minVal, maxVal←read from inFile

          zeroFramedAry←dynamically allocate.

          newLabel←0

step 1:    zero2D (zeroFramedAry)

step 2:   loadImage (inFile, zeroFramedAry)

step 3:   if connectness == 4
                  connected4 (zeroFramedAry, newLabel, EQTable, prettyPrintFile, logFile)

step 4:   if connectness == 8
                  connected8 (zeroFramedAry, newLabel, EQTable, prettyPrintFile, logFile)

step 5:   labelFile←output numRows, numCols, newMin, newMax to labelFile

step 6: printImg (zeroFramedAry, labelFile) // Output the result of pass3 inside of zeroFramedAry.

step 7: printCCproperty (propertyFile) // print cc properties to propertyFile

step 8: drawBoxes (zeroFramedAry, CCproperty, trueNumCC, logFile) // draw on zeroFramed image.

step 9: prettyDotPrint (zeroFramedAry, prettyPrintFile)

step 10: prettyPrintFile←print trueNumCC to prettyPrintFile with proper caption

step 12: close all files
```

Source code

```
import java.io.*;
import java.util.StringTokenizer;

class Property{
    public int label;
    public int numPixels;
    public int minR;
    public int minC;
    public int maxR;
    public int maxC;
}

class ccLabel{
    public int numRows;
    public int numCols;
    public int minVal;
    public int maxVal;
    public int newLabel;
    public int trueNumCC;
    public int newMin;
    public int newMax;
    public int[][] zeroFramedAry;
    public int[] nonZeroNeighborAry = new int[5];
    public int[] EQTable;
    public int[] property;

    public void zero2D(int[][] array){
        for(int i = 0; i < array.length; i++){
            for(int j = 0; j < array[i].length; j++){
                array[i][j] = 0;
            }
        }
    }

    public void loadImage(BufferedReader inFile, int[][] zeroFramedAry) throws IOException {
        for (int i = 1; i < zeroFramedAry.length - 1; i++) {
            String currentLine = inFile.readLine();
            StringTokenizer currentLineTokenizer = new StringTokenizer(currentLine);
            for (int j = 1; j < zeroFramedAry[i].length - 1; j++) {
                if (currentLineTokenizer.hasMoreTokens()) {
                    zeroFramedAry[i][j] = Integer.parseInt(currentLineTokenizer.nextToken());
                }
            }
        }
    }

    public void prettyDotPrint(int[][] zeroFramedAry, BufferedWriter prettyPrintFile) throws IOException {
        int cellWidth = Integer.toString(maxVal).length() + 2; // Fixed width for each cell, based on maxVal

        for (int i = 0; i < zeroFramedAry.length; i++) {
            for (int j = 0; j < zeroFramedAry[i].length; j++) {
                if (zeroFramedAry[i][j] != 0) {
                    prettyPrintFile.write(String.format("%" + cellWidth + "d", zeroFramedAry[i][j])); // Print value
with padding
                } else {
                    prettyPrintFile.write(String.format("%" + cellWidth + "s", ".")); // Print dot with padding
                }
            }
            prettyPrintFile.write("\n"); // New line at the end of each row
        }
    }


    public void connected4(int[][] zeroFramedAry, int newLabel, int[] EQTable, BufferedWriter prettyPrintFile,
BufferedWriter logFile) throws IOException {
        logFile.write("entering connected4 method" + "\n");
        connected4Pass1(zeroFramedAry, newLabel, EQTable, logFile);
        logFile.write("After connected4 pass1, newLabel = " + newLabel + "\n");
        prettyDotPrint(zeroFramedAry, prettyPrintFile);
        printEQTable(newLabel, prettyPrintFile);

        connected4Pass2(zeroFramedAry, newLabel, EQTable, logFile);
        logFile.write("After connected4 pass2, newLabel = " + newLabel + "\n");
        prettyDotPrint(zeroFramedAry, prettyPrintFile);
        printEQTable(newLabel, prettyPrintFile);

//        trueNumCC = manageEQTable(EQTable, newLabel);
//        printEQTable(newLabel, prettyPrintFile);
//        newMin = 0;
//        newMax = trueNumCC;
//        property = new int[trueNumCC + 1];
//        logFile.write("In connected4, after manage EQAry, trueNumCC = " + trueNumCC);

//        Property[] properties = new Property[(numRows*numCols/4)];
//       connected4Pass3(zeroFramedAry, EQTable, properties, trueNumCC, logFile);

        prettyDotPrint(zeroFramedAry, prettyPrintFile);

        printEQTable(newLabel, prettyPrintFile);

        logFile.write("Leaving connected4 method");
    }

    public void connected4Pass1(int[][] zeroFramedAry, int newLabel, int[] EQTable, BufferedWriter logFile)
throws IOException {
        logFile.write("Entering connected4Pass1()");
        for (int i = 1; i < zeroFramedAry.length - 1; i++) {
            for (int j = 1; j < zeroFramedAry[i].length - 1; j++) {
                if (zeroFramedAry[i][j] > 0) {
                    // Case 1
                    if (zeroFramedAry[i - 1][j] == 0 && zeroFramedAry[i][j - 1] == 0) {
                        newLabel++;
                        zeroFramedAry[i][j] = newLabel;
                    }
                    // Case 2
                    else if (zeroFramedAry[i - 1][j] != 0 && zeroFramedAry[i - 1][j] == zeroFramedAry[i][j - 1]) {
                        zeroFramedAry[i][j] = zeroFramedAry[i - 1][j];
                    }
                    // Case 3: Conflict case, update EQTable
                    else if (zeroFramedAry[i - 1][j] != zeroFramedAry[i][j - 1] &&
                            (zeroFramedAry[i - 1][j] != 0 || zeroFramedAry[i][j - 1] != 0)) {
                        if (zeroFramedAry[i - 1][j] == 0) {
                            zeroFramedAry[i][j] = zeroFramedAry[i][j - 1];
                        } else if (zeroFramedAry[i][j - 1] == 0) {
                            zeroFramedAry[i][j] = zeroFramedAry[i - 1][j];
                        } else {
                            int minLabel = Math.min(zeroFramedAry[i - 1][j], zeroFramedAry[i][j - 1]);
                            int maxLabel = Math.max(zeroFramedAry[i - 1][j], zeroFramedAry[i][j - 1]);
                            zeroFramedAry[i][j] = minLabel;
                            EQTable[maxLabel] = minLabel;
                            System.out.println("Updated EQTable: " + maxLabel + " -> " + minLabel);
                        }
                        System.out.println(newLabel);
                    }
                }
            }
        }
        logFile.write("Leaving connected4Pass1()");
```

```
   }


   public void connected4Pass2(int[][] zeroFramedAry, int newLabel, int[] EQTable, BufferedWriter logFile)
throws IOException {
       logFile.write("Entering connected4Pass2()");

       for (int i = zeroFramedAry.length - 1; i > 0; i--) {
           for (int j = zeroFramedAry[i].length - 1; j > 0; j--) {
               if (zeroFramedAry[i][j] > 0) {
                   int p = zeroFramedAry[i][j];
                   int e = zeroFramedAry[i][j + 1];
                   int g = zeroFramedAry[i + 1][j];


                   if ((p != 0 || e != 0 || g != 0) && p != e && p != g && e != g) {
                       int min = Integer.MAX_VALUE;


                       if (p != 0) min = p;
                       if (e != 0 && e < min) min = e;
                       if (g != 0 && g < min) min = g;


                       zeroFramedAry[i][j] = min;


                       if (e != 0 && e != min) {
                           EQTable[Math.max(p, e)] = min;
                       }
                       if (g != 0 && g != min) {
                           EQTable[Math.max(p, g)] = min;
                       }
                   }
               }
           }
       }
       logFile.write("Leaving connected4Pass2()");
   }

   public void connected4Pass3(int[][] zeroFramedAry, int[] EQTable, Property[] CCproperty, int
trueNumCC, BufferedWriter logFile) throws IOException {
       logFile.write("Entering connectPass3 method\n");
```

```
    for (int i = 1; i <= trueNumCC; i++) {
        CCproperty[i] = new Property();
        CCproperty[i].label = i;
        CCproperty[i].numPixels = 0;
        CCproperty[i].minR = numRows;
        CCproperty[i].maxR = 0;
        CCproperty[i].minC = numCols;
        CCproperty[i].maxC = 0;
    }


    for (int r = 1; r < zeroFramedAry.length - 1; r++) {
        for (int c = 1; c < zeroFramedAry[r].length - 1; c++) {

            if (zeroFramedAry[r][c] > 0) {

                int k = EQTable[zeroFramedAry[r][c]];
                zeroFramedAry[r][c] = k;


                CCproperty[k].numPixels++;
                if (r < CCproperty[k].minR) {
                    CCproperty[k].minR = r;
                }
                if (r > CCproperty[k].maxR) {
                    CCproperty[k].maxR = r;
                }
                if (c < CCproperty[k].minC) {
                    CCproperty[k].minC = c;
                }
                if (c > CCproperty[k].maxC) {
                    CCproperty[k].maxC = c;
                }
            }
        }
    }

    logFile.write("Leaving connectPass3 method\n");
}

public void printImg(int[][] zeroFramedAry, BufferedWriter labelFile) throws IOException {
    for(int i = 0; i < zeroFramedAry.length; i++){
        for(int j = 0; j < zeroFramedAry[i].length; j++){
            labelFile.write(Integer.toString(zeroFramedAry[i][j]) + " ");
            }
            labelFile.write("\n");
        }
    }


    public void printEQTable(int newLabel, BufferedWriter prettyPrintFile) throws IOException {
        prettyPrintFile.write("Equivalence Table (up to newLabel " + newLabel + "):\n");
        for (int i = 1; i <= EQTable.length-1; i++) {
            if (EQTable[i] != 0) {
                prettyPrintFile.write(EQTable[i] + " ");
            }
        }
        prettyPrintFile.write("\n");
    }



}

public class YournetF_Project6_Main {
    public static void main(String[] args) throws IOException {

        //Checks to see if the inFile can be read.
        BufferedReader inFileReader = null;
        try{
            inFileReader = new BufferedReader(new FileReader(args[0]));
        } catch (FileNotFoundException e) {
            System.out.println("Unable to open file '" + args[0] + "'");
        }

        //Checks to see if the prettyPrintFile can be opened.
        BufferedWriter prettyPrintFile = null;
        try{
            prettyPrintFile = new BufferedWriter(new FileWriter(args[2]));
        } catch (FileNotFoundException e) {
            System.out.println("Unable to open file '" + args[2] + "'");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        //Checks to see if the labelFile can be opened.
        BufferedWriter labelFile = null;
        try{
```

```
    labelFile = new BufferedWriter(new FileWriter(args[3]));
} catch (FileNotFoundException e) {
    System.out.println("Unable to open file '" + args[3] + "'");
} catch (IOException e) {
    throw new RuntimeException(e);
}

//Checks to see if the propertyFile can be opened.
BufferedWriter propertyFile = null;
try{
    propertyFile = new BufferedWriter(new FileWriter(args[4]));
} catch (FileNotFoundException e) {
    System.out.println("Unable to open file '" + args[4] + "'");
} catch (IOException e) {
    throw new RuntimeException(e);
}

//Checks to see if the logFile can be opened.
BufferedWriter logFile = null;
try{
    logFile = new BufferedWriter(new FileWriter(args[5]));
} catch (FileNotFoundException e) {
    System.out.println("Unable to open file '" + args[5] + "'");
} catch (IOException e) {
    throw new RuntimeException(e);
}


//Attempts to read the header of the inFile.
String inFileHeader = null;
try {
    assert inFileReader != null;
    inFileHeader = inFileReader.readLine();
} catch (IOException e) {
    throw new RuntimeException(e);
}

//Checks the header and assigns the proper values to the Morphology class.
StringTokenizer inFileTokenizer = new StringTokenizer(inFileHeader);
int numImgRows = Integer.parseInt(inFileTokenizer.nextToken());
int numImgCols = Integer.parseInt(inFileTokenizer.nextToken());
int imgMin = Integer.parseInt(inFileTokenizer.nextToken());
int imgMax = Integer.parseInt(inFileTokenizer.nextToken());
```

```
}



        int connectedness = Integer.parseInt(args[1]);

        ccLabel ccInstance = new ccLabel();

        ccInstance.numRows = numImgRows;
        ccInstance.numCols = numImgCols;
        ccInstance.minVal = imgMin;
        ccInstance.maxVal = imgMax;

        ccInstance.EQTable = new int[(ccInstance.numRows * numImgCols)/4];

        ccInstance.zeroFramedAry = new int[ccInstance.numRows + 2][ccInstance.numCols + 2];

        ccInstance.newLabel = 0;

        ccInstance.zero2D(ccInstance.zeroFramedAry);

        ccInstance.loadImage(inFileReader, ccInstance.zeroFramedAry);

        if(connectedness == 4){
            ccInstance.connected4(ccInstance.zeroFramedAry, ccInstance.newLabel, ccInstance.EQTable,
prettyPrintFile, logFile);
        }
        else if(connectedness == 8){
            //call connected8()
        }

        assert labelFile != null;
        labelFile.write(numImgRows + " " +  numImgCols + " " +  ccInstance.newMin + " " +
ccInstance.newMax);
        ccInstance.printImg(ccInstance.zeroFramedAry, labelFile);

        prettyPrintFile.close();
        inFileReader.close();
        logFile.close();
        propertyFile.close();
        labelFile.close();




    }
}
```

PrettyPrintFile for data1

```
 .  .  .  .  .  .  .  .  .  .  .  .
 .  1  1  .  2  .  .  3  .  4  .  .
 .  .  1  .  2  2  .  3  .  4  .  .
 .  .  1  .  .  2  .  3  .  4  .  .
 .  5  1  .  .  2  .  3  .  4  4  .
 .  5  .  6  6  .  .  3  .  4  .  .
 .  .  .  .  .  7  7  3  3  3  .  .
 .  .  .  8  .  .  .  .  3  .  9  .
 . 10 10  8  8  .  . 11  . 12  .  .
 . 10  .  8  . 13 13 11 11  .  .  .
 .  .  .  .  .  . 13  . 11  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .
Equivalence Table (up to newLabel 0):
3 1 3 8 11
 .  .  .  .  .  .  .  .  .  .  .  .
 .  1  1  .  2  .  .  3  .  4  .  .
 .  .  1  .  2  2  .  3  .  4  .  .
 .  .  1  .  .  2  .  3  .  4  .  .
 .  5  1  .  .  2  .  3  .  4  4  .
 .  5  .  6  6  .  .  3  .  3  .  .
 .  .  .  .  .  3  3  3  3  3  .  .
 .  .  .  8  .  .  .  .  3  .  9  .
 . 10  8  8  8  .  . 11  . 12  .  .
 . 10  .  8  . 13 13 11 11  .  .  .
 .  .  .  .  .  . 13  . 11  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .
Equivalence Table (up to newLabel 0):
3 1 3 8 11
 .  .  .  .  .  .  .  .  .  .  .  .
 .  1  1  .  2  .  .  3  .  4  .  .
 .  .  1  .  2  2  .  3  .  4  .  .
 .  .  1  .  .  2  .  3  .  4  .  .
 .  5  1  .  .  2  .  3  .  4  4  .
 .  5  .  6  6  .  .  3  .  3  .  .
 .  .  .  .  .  3  3  3  3  3  .  .
 .  .  .  8  .  .  .  .  3  .  9  .
 . 10  8  8  8  .  . 11  . 12  .  .
 . 10  .  8  . 13 13 11 11  .  .  .
 .  .  .  .  .  . 13  . 11  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .
Equivalence Table (up to newLabel 0):
3 1 3 8 11
```

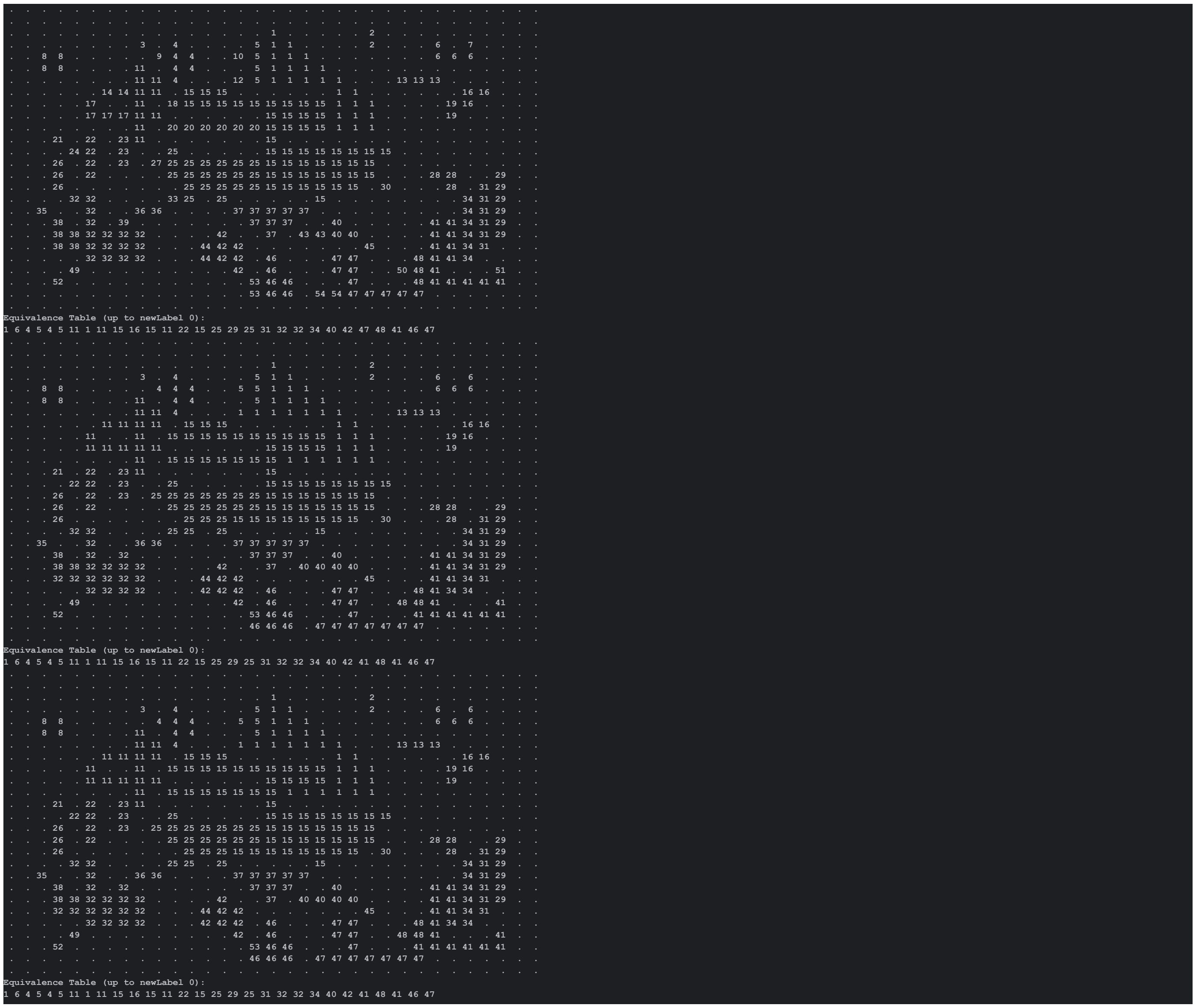
LabelFile for data1

```
10 10 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 2 0 0 3 0 4 0 0
0 0 1 0 2 2 0 3 0 4 0 0
0 0 1 0 0 2 0 3 0 4 0 0
0 5 1 0 0 2 0 3 0 4 4 0
0 5 0 6 6 0 0 3 0 3 0 0
0 0 0 0 0 3 3 3 3 3 0 0
0 0 0 8 0 0 0 0 3 0 9 0
0 10 8 8 8 0 0 11 0 12 0 0
0 10 0 8 0 13 13 11 11 0 0 0
0 0 0 0 0 0 13 0 11 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
```

Logfile for data 1

```
entering connected4 method
Entering connected4Pass1()Leaving connected4Pass1()After connected4 pass1, newLabel = 0
Entering connected4Pass2()Leaving connected4Pass2()After connected4 pass2, newLabel = 0
Leaving connected4 method
```

PrettyPrint for data2



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .  .  .  .  .  2  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  3  .  4  .  .  .  .  5  1  1  .  .  .  .  2  .  .  .  6  .  7  .  .  .  .
 .  .  8  8  .  .  .  .  .  9  4  4  .  . 10  5  1  1  1  .  .  .  .  .  .  .  6  6  6  .  .  .  .
 .  .  8  8  .  .  .  . 11  .  4  4  .  .  .  5  1  1  1  1  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  . 11 11  4  .  .  . 12  5  1  1  1  1  1  .  .  . 13 13 13  .  .  .  .  .  .
 .  .  .  .  .  . 14 14 11 11  . 15 15 15  .  .  .  .  .  .  1  1  .  .  .  .  .  . 16 16  .  .  .
 .  .  .  .  . 17  .  . 11  . 18 15 15 15 15 15 15 15 15 15  1  1  1  .  .  .  . 19 16  .  .  .  .
 .  .  .  .  . 17 17 17 11 11  .  .  .  .  .  . 15 15 15 15  1  1  1  .  .  .  . 19  .  .  .  .  .
 .  .  .  .  .  .  .  . 11  . 20 20 20 20 20 20 15 15 15 15  1  1  1  .  .  .  .  .  .  .  .  .  .
 .  .  . 21  . 22  . 23 11  .  .  .  .  .  .  . 15  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  . 24 22  . 23  .  . 25  .  .  .  .  . 15 15 15 15 15 15 15 15  .  .  .  .  .  .  .  .  .
 .  .  . 26  . 22  . 23  . 27 25 25 25 25 25 25 15 15 15 15 15 15 15  .  .  .  .  .  .  .  .  .  .
 .  .  . 26  . 22  .  .  .  . 25 25 25 25 25 25 15 15 15 15 15 15 15  .  .  . 28 28  .  . 29  .  .
 .  .  . 26  .  .  .  .  .  .  . 25 25 25 25 25 15 15 15 15 15 15  . 30  .  .  . 28  . 31 29  .  .
 .  .  .  . 32 32  .  .  .  . 33 25  . 25  .  .  .  .  . 15  .  .  .  .  .  .  .  . 34 31 29  .  .
 .  . 35  .  . 32  .  . 36 36  .  .  .  . 37 37 37 37 37  .  .  .  .  .  .  .  .  . 34 31 29  .  .
 .  .  . 38  . 32  . 39  .  .  .  .  .  .  . 37 37 37  .  . 40  .  .  .  .  . 41 41 34 31 29  .  .
 .  .  . 38 38 32 32 32 32  .  .  .  . 42  .  . 37  . 43 43 40 40  .  .  .  . 41 41 34 31 29  .  .
 .  .  . 38 38 32 32 32 32  .  .  . 44 42 42  .  .  .  .  .  .  . 45  .  .  . 41 41 34 31  .  .  .
 .  .  .  .  . 32 32 32 32  .  .  . 44 42 42  . 46  .  .  . 47 47  .  .  . 48 41 41 34  .  .  .  .
 .  .  .  . 49  .  .  .  .  .  .  .  .  . 42  . 46  .  .  . 47 47  .  . 50 48 41  .  .  . 51  .  .
 .  .  . 52  .  .  .  .  .  .  .  .  .  .  . 53 46 46  .  .  . 47  .  .  . 48 41 41 41 41 41  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 53 46 46  . 54 54 47 47 47 47 47  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Equivalence Table (up to newLabel 0):
1 6 4 5 4 5 11 1 11 15 16 15 11 22 15 25 29 25 31 32 32 34 40 42 47 48 41 46 47
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .  .  .  .  .  2  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  3  .  4  .  .  .  .  5  1  1  .  .  .  .  2  .  .  .  6  .  6  .  .  .  .
 .  .  8  8  .  .  .  .  .  4  4  4  .  .  5  5  1  1  1  .  .  .  .  .  .  .  6  6  6  .  .  .  .
 .  .  8  8  .  .  .  . 11  .  4  4  .  .  .  5  1  1  1  1  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  . 11 11  4  .  .  .  1  1  1  1  1  1  1  .  .  . 13 13 13  .  .  .  .  .  .
 .  .  .  .  .  . 11 11 11 11  . 15 15 15  .  .  .  .  .  .  1  1  .  .  .  .  .  . 16 16  .  .  .
 .  .  .  .  . 11  .  . 11  . 15 15 15 15 15 15 15 15 15 15  1  1  1  .  .  .  . 19 16  .  .  .  .
 .  .  .  .  . 11 11 11 11 11  .  .  .  .  .  . 15 15 15 15  1  1  1  .  .  .  . 19  .  .  .  .  .
 .  .  .  .  .  .  .  . 11  . 15 15 15 15 15 15 15  1  1  1  1  1  1  .  .  .  .  .  .  .  .  .  .
 .  .  . 21  . 22  . 23 11  .  .  .  .  .  .  . 15  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  . 22 22  . 23  .  . 25  .  .  .  .  . 15 15 15 15 15 15 15 15  .  .  .  .  .  .  .  .  .
 .  .  . 26  . 22  . 23  . 25 25 25 25 25 25 25 15 15 15 15 15 15 15  .  .  .  .  .  .  .  .  .  .
 .  .  . 26  . 22  .  .  .  . 25 25 25 25 25 25 15 15 15 15 15 15 15  .  .  . 28 28  .  . 29  .  .
 .  .  . 26  .  .  .  .  .  .  . 25 25 25 15 15 15 15 15 15 15 15  . 30  .  .  . 28  . 31 29  .  .
 .  .  .  . 32 32  .  .  .  . 25 25  . 25  .  .  .  .  . 15  .  .  .  .  .  .  .  . 34 31 29  .  .
 .  . 35  .  . 32  .  . 36 36  .  .  .  . 37 37 37 37 37  .  .  .  .  .  .  .  .  . 34 31 29  .  .
 .  .  . 38  . 32  . 32  .  .  .  .  .  .  . 37 37 37  .  . 40  .  .  .  .  . 41 41 34 31 29  .  .
 .  .  . 38 38 32 32 32 32  .  .  .  . 42  .  . 37  . 40 40 40 40  .  .  .  . 41 41 34 31 29  .  .
 .  .  . 32 32 32 32 32 32  .  .  . 44 42 42  .  .  .  .  .  .  . 45  .  .  . 41 41 34 31  .  .  .
 .  .  .  .  . 32 32 32 32  .  .  . 42 42 42  . 46  .  .  . 47 47  .  .  . 48 41 34 34  .  .  .  .
 .  .  .  . 49  .  .  .  .  .  .  .  .  . 42  . 46  .  .  . 47 47  .  . 48 48 41  .  .  . 41  .  .
 .  .  . 52  .  .  .  .  .  .  .  .  .  .  . 53 46 46  .  .  . 47  .  .  . 41 41 41 41 41 41  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 46 46 46  . 47 47 47 47 47 47 47  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Equivalence Table (up to newLabel 0):
1 6 4 5 4 5 11 1 11 15 16 15 11 22 15 25 29 25 31 32 32 34 40 42 41 48 41 46 47
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .  .  .  .  .  2  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  3  .  4  .  .  .  .  5  1  1  .  .  .  .  2  .  .  .  6  .  6  .  .  .  .
 .  .  8  8  .  .  .  .  .  4  4  4  .  .  5  5  1  1  1  .  .  .  .  .  .  .  6  6  6  .  .  .  .
 .  .  8  8  .  .  .  . 11  .  4  4  .  .  .  5  1  1  1  1  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  . 11 11  4  .  .  .  1  1  1  1  1  1  1  .  .  . 13 13 13  .  .  .  .  .  .
 .  .  .  .  .  . 11 11 11 11  . 15 15 15  .  .  .  .  .  .  1  1  .  .  .  .  .  . 16 16  .  .  .
 .  .  .  .  . 11  .  . 11  . 15 15 15 15 15 15 15 15 15 15  1  1  1  .  .  .  . 19 16  .  .  .  .
 .  .  .  .  . 11 11 11 11 11  .  .  .  .  .  . 15 15 15 15  1  1  1  .  .  .  . 19  .  .  .  .  .
 .  .  .  .  .  .  .  . 11  . 15 15 15 15 15 15 15  1  1  1  1  1  1  .  .  .  .  .  .  .  .  .  .
 .  .  . 21  . 22  . 23 11  .  .  .  .  .  .  . 15  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 .  .  .  . 22 22  . 23  .  . 25  .  .  .  .  . 15 15 15 15 15 15 15 15  .  .  .  .  .  .  .  .  .
 .  .  . 26  . 22  . 23  . 25 25 25 25 25 25 25 15 15 15 15 15 15 15  .  .  .  .  .  .  .  .  .  .
 .  .  . 26  . 22  .  .  .  . 25 25 25 25 25 25 15 15 15 15 15 15 15  .  .  . 28 28  .  . 29  .  .
 .  .  . 26  .  .  .  .  .  .  . 25 25 25 15 15 15 15 15 15 15 15  . 30  .  .  . 28  . 31 29  .  .
 .  .  .  . 32 32  .  .  .  . 25 25  . 25  .  .  .  .  . 15  .  .  .  .  .  .  .  . 34 31 29  .  .
 .  . 35  .  . 32  .  . 36 36  .  .  .  . 37 37 37 37 37  .  .  .  .  .  .  .  .  . 34 31 29  .  .
 .  .  . 38  . 32  . 32  .  .  .  .  .  .  . 37 37 37  .  . 40  .  .  .  .  . 41 41 34 31 29  .  .
 .  .  . 38 38 32 32 32 32  .  .  .  . 42  .  . 37  . 40 40 40 40  .  .  .  . 41 41 34 31 29  .  .
 .  .  . 32 32 32 32 32 32  .  .  . 44 42 42  .  .  .  .  .  .  . 45  .  .  . 41 41 34 31  .  .  .
 .  .  .  .  . 32 32 32 32  .  .  . 42 42 42  . 46  .  .  . 47 47  .  .  . 48 41 34 34  .  .  .  .
 .  .  .  . 49  .  .  .  .  .  .  .  .  . 42  . 46  .  .  . 47 47  .  . 48 48 41  .  .  . 41  .  .
 .  .  . 52  .  .  .  .  .  .  .  .  .  .  . 53 46 46  .  .  . 47  .  .  . 41 41 41 41 41 41  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  . 46 46 46  . 47 47 47 47 47 47 47  .  .  .  .  .  .  .
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Equivalence Table (up to newLabel 0):
1 6 4 5 4 5 11 1 11 15 16 15 11 22 15 25 29 25 31 32 32 34 40 42 41 48 41 46 47
```

LabelFile for data2

```
24 31 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 0 0 1 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 3 0 4 0 0 0 0 5 1 1 0 0 0 0 2 0 0 0 6 0 6 0 0 0 0
0 0 8 8 0 0 0 0 0 4 4 4 0 0 5 5 1 1 1 0 0 0 0 0 0 0 6 6 6 0 0 0 0
0 0 8 8 0 0 0 0 11 0 4 4 0 0 0 5 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 11 11 4 0 0 0 1 1 1 1 1 1 1 0 0 0 13 13 13 0 0 0 0 0 0
0 0 0 0 0 0 11 11 11 11 0 15 15 15 0 0 0 0 0 1 1 0 0 0 0 0 0 16 16 0 0 0
0 0 0 0 0 11 0 0 11 0 15 15 15 15 15 15 15 15 15 15 1 1 1 0 0 0 0 19 16 0 0 0 0
0 0 0 0 0 11 11 11 11 11 0 0 0 0 0 0 15 15 15 15 1 1 1 0 0 0 0 19 0 0 0 0 0
0 0 0 0 0 0 0 0 11 0 15 15 15 15 15 15 15 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 21 0 22 0 23 11 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 22 22 0 23 0 0 25 0 0 0 0 0 15 15 15 15 15 15 15 15 0 0 0 0 0 0 0 0 0
0 0 0 26 0 22 0 23 0 25 25 25 25 25 25 25 15 15 15 15 15 15 15 0 0 0 0 0 0 0 0 0 0
0 0 0 26 0 22 0 0 0 0 25 25 25 25 25 25 15 15 15 15 15 15 15 0 0 0 28 28 0 0 29 0 0
0 0 0 26 0 0 0 0 0 0 0 25 25 25 15 15 15 15 15 15 15 15 0 30 0 0 0 28 0 31 29 0 0
0 0 0 0 32 32 0 0 0 0 25 25 0 25 0 0 0 0 0 15 0 0 0 0 0 0 0 0 34 31 29 0 0
0 0 35 0 0 32 0 0 36 36 0 0 0 0 37 37 37 37 37 0 0 0 0 0 0 0 0 0 34 31 29 0 0
0 0 0 38 0 32 0 32 0 0 0 0 0 0 0 37 37 37 0 0 40 0 0 0 0 0 41 41 34 31 29 0 0
0 0 0 38 38 32 32 32 32 0 0 0 0 42 0 0 37 0 40 40 40 40 0 0 0 0 41 41 34 31 29 0 0
0 0 0 32 32 32 32 32 32 0 0 0 44 42 42 0 0 0 0 0 0 0 45 0 0 0 41 41 34 31 0 0 0
0 0 0 0 0 32 32 32 32 0 0 0 42 42 42 0 46 0 0 0 47 47 0 0 0 48 41 34 34 0 0 0 0
0 0 0 0 49 0 0 0 0 0 0 0 0 0 42 0 46 0 0 0 47 47 0 0 48 48 41 0 0 0 41 0 0
0 0 0 52 0 0 0 0 0 0 0 0 0 0 0 53 46 46 0 0 0 47 0 0 0 41 41 41 41 41 41 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 46 46 0 47 47 47 47 47 47 47 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
```

Logfile for data2

```
entering connected4 method
```

```
Entering connected4Pass1()Leaving connected4Pass1()After connected4 pass1, newLabel = 0
Entering connected4Pass2()Leaving connected4Pass2()After connected4 pass2, newLabel = 0
Leaving connected4 method
```