**CSCI 381 – Computer Vision (C++)**

**Program: Project 7: ChainCode**

**Name: Isaac Gordon**

**Due Date:**

**Soft copy: 4/31/2019 Tuesday before midnight**

**Hard copy: 5/02/2019 Thursday in class**

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

IV. main (...)

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

step 0: inFile 🡨 open input file

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate imgAry size of numRows by numCols

dynamically allocate ZeroFramedAry size of numRows+2 by numCo+2

Step 1: chainCodeFileName 🡨 argv[1] + “\_chainCode.txt”

Step 2: chainCodeFile 🡨 open (chainCodeFileName)

Step 3: loadImage (inFile, ZeroFramedAry) // begins at ZeroFramedAry (1,1)

Step 4: getChainCode (ZeroFramedAry, chainCodeFile)

Step 5: close (chainCodeFile)

Step 6: re-open (chainCodeFile)

Step 7: deCompressFileName 🡨 argv[1] + “\_chainCodeDecompressed.txt”

Step 8: deCompressFile 🡨 open (deCompressFileName)

Step 9: reconstructObject (chainCodeFile, deCompressFile, imgAry)

Step 10: close all files

**Code**

#include<iostream>

#include<fstream>

#include<string>

#include<cstring>

#include<cmath>

using namespace std;

/\* Class Definitions \*/

class Image{

public:

//data structures

int numRows, numCols, minVal, maxVal, label, numBoundryPts;

int\*\* imgAry;

int\*\* zeroFramedAry;

//Functions

Image();

Image(ifstream &imgFileInputStream);

void setLabel(int i){

label = i;

}

void setNumRows(int i){

numRows = i;

}

void setNumCols(int i){

numCols = i;

}

void setMinVal(int i){

minVal = i;

}

void setMaxVal(int i ){

maxVal = i;

}

void setNumBoundryPts(int i){

numBoundryPts = i;

}

void setZeroFrame(int \*\*&ary){

ary = new int\*[numRows + 2];

for(int i = 0; i < numRows + 2; i++){

ary[i] = new int[numCols + 2];

for(int j = 0; j < numCols + 2; j++){

ary[i][j] = 0;

}//for

}//for

}//setZeroFrame

void loadImage(ifstream &imgFileInputStream, int \*\*&ary){

setZeroFrame(ary);

for(int i = 0; i < numRows; i++){

for(int j = 0; j < numCols; j++){

imgFileInputStream >> ary[i+1][j+1];

if(ary[i+1][j+1] != 0) label = ary[i+1][j+1];

}//for

}//for

}//loadImage

// Image operator=(const Image& im){

// // int\*\* ary;

// // int\*\* zeroAry;

// // Image img;

// // img.setLabel(im.label);

// // img.setNumRows(im.numRows);

// // img.setNumCols(im.numCols);

// // img.setMinVal(im.minVal);

// // img.setMaxVal(im.maxVal);

// // img.setNumBoundryPts(im.numBoundryPts);

// // zeroAry = new int\*[im.numRows + 2];

// // int\*\* imz = im.zeroFramedAry;

// // for(int a = 0; a < im.numRows + 2; a++){

// // zeroAry[a] = new int[im.numCols + 2];

// // for(int b = 0; b < im.numCols + 2; b++){

// // zeroAry[a][b] = imz[a][b];

// // }//for

// // }//for

// // ary = new int\*[im.numRows];

// // for(int a = 0; a < im.numRows; a++){

// // ary[a] = new int[im.numCols];

// // for(int b = 0; b < im.numCols; b++){

// // ary[a][b] = im.imgAry[a][b];

// // }//for

// // }//for

// // img.imgAry = ary;

// // img.zeroFramedAry = zeroAry;

// return im;

// }

};//Image Class

class ChainCode{

public:

//class declarations

class Point{

public: int row; int col;

void setRow(int i){row = i;}

void setCol(int i){col = i;}

int getRow(){return row;}

int getCol(){return col;}

Point(){};

Point(int r, int c){

this->row = r;

this->col = c;

}//contructor

//overoad operators

bool operator!=(const Point& p){

if(row == p.row )

if(col == p.col)

return false;

return true;

}//not-equal

bool operator==(const Point& p){

if(row == p.row )

if(col == p.col)

return true;

return false;

}//is-equal

// Point operator=(const Point& p){

// int r = p.row;

// int c = p.col;

// Point a(r,c);

// return a;

// }//copy constructor

};//Point Class

friend class Image;

//data structs

Point startP, currentP, nextP;

Point neighborhood[8];

int lastQ, chainDir;

int compassTable[8];

Image image;

//Functions

ChainCode(){

compassTable[0] = 6;

compassTable[1] = 0;

compassTable[2] = 0;

compassTable[3] = 2;

compassTable[4] = 2;

compassTable[5] = 4;

compassTable[6] = 4;

compassTable[7] = 6;

}

void getChainCode(Image img, ofstream &chainCodeFile){

//STEP 0

image = img;

chainCodeFile << image.numRows << " ";

chainCodeFile << image.numCols << " ";

chainCodeFile << image.minVal << " ";

chainCodeFile << image.maxVal << " " << endl;

image.setNumBoundryPts(0);

//STEP 1 and 2

int\*\* ary = image.zeroFramedAry;

bool found = false;

for(int i = 1; i < image.numRows + 1; i++){

for(int j = 1; j < image.numCols; j++){

if(ary[i][j] > 0){

chainCodeFile << (i-1) << " ";

chainCodeFile << (j-1) << " ";

chainCodeFile << ary[i][j] << " ";

image.setLabel(ary[i][j]);

Point kkk(i,j);

currentP = kkk;

startP = kkk;

lastQ = 4;

found = true;

break;

}

}//for cols

if(found) break;

}//for rows

lastQ = (lastQ + 1) % 8;

do {

//STEP 3

//STEP 4

chainDir = findNextP(lastQ, currentP);

image.setNumBoundryPts(image.numBoundryPts + 1);

//STEP 5

chainCodeFile << chainDir << " ";

//STEP 6

nextP = neighborhood[chainDir];

currentP = nextP; //i feel that this and the previous line should be swapped

lastQ = compassTable[chainDir];

} while(currentP.row != startP.row || currentP.col != startP.col);

//STEP 7

cout << "The total number of boundry pixels: " << image.numBoundryPts << endl;

}//getChainCode

int findNextP(int lastD, Point currentPoint){

int cd;

loadNeighborsCoord(currentPoint);

int index = lastD;

bool found = false;

int\*\* ary = image.zeroFramedAry;

int loopCounter = 0;

while(!found && loopCounter < 20){

int iRow = (neighborhood[index]).row;

int jCol = (neighborhood[index]).col;

if(ary[iRow][jCol] == image.label){

cd = index;

found = true;

}

index = (index+1) % 8;

loopCounter++;

}//while

if(loopCounter >= 20){

cout << "crazy loop at 225" << endl;

}

return cd;

}//findNextP

void loadNeighborsCoord(Point currentP){

int r = currentP.getRow();

int c = currentP.getCol();

int index = 0;

int nexti[8] = {3,2,1,4,0,5,6,7};

for(int i = r-1; i <= r+1; i++){

for(int j = c-1; j <= c+1; j++){

if(i==r && j==c) continue;

else{

Point p(i,j);

neighborhood[nexti[index]] = p;

index++;

}//else

}//for

}//for

}//loadNeighborsCoord

void reconstructObject(ifstream &chainCodeFile, ofstream &boundaryFile ,ofstream &deCompressFile, int \*\*&imgAry){

int cRows, cCols, cMinVal, cMaxVal;

int clabel, tmpLabel, startRow, startCol;

//get cahin code header info

chainCodeFile >> cRows;

chainCodeFile >> cCols;

chainCodeFile >> cMinVal;

chainCodeFile >> cMaxVal;

chainCodeFile >> startRow;

chainCodeFile >> startCol;

chainCodeFile >> clabel;

//set start values

tmpLabel = clabel + 2;

startP = Point(startRow, startCol);

cout << "startP for decom is P(" << startRow << ", " << startCol << ")\n";

//dynamically allocate imgAry

imgAry = new int\*[cRows];

for(int i = 0; i < cRows; i++){

imgAry[i] = new int[cCols];

for(int j = 0; j < cCols; j++){

imgAry[i][j] = 0;

}//for

}//for

//output headers to files

deCompressFile << cRows << " ";

deCompressFile << cCols << " ";

deCompressFile << cMinVal << " ";

deCompressFile << cMaxVal << endl;

boundaryFile << cRows << " ";

boundaryFile << cCols << " ";

boundaryFile << cMinVal << " ";

boundaryFile << cMaxVal << endl;

//construct boundary and output boundary to boundaryFile

constructBoundary(chainCodeFile, boundaryFile, imgAry, tmpLabel);

printImageAryToFile(imgAry, cRows, cCols, boundaryFile);

//fill interior of the object then print the image to decompressedFile

fillInterior(imgAry, cRows, cCols, tmpLabel);

printImageAryToFile(imgAry, cRows, cCols, deCompressFile);

}//reconstructObject

void constructBoundary(ifstream &chainCodeFile, ofstream &boundaryFile, int \*\*&imgAry, int tmplabel){

imgAry[startP.row][startP.col] = tmplabel;

currentP = startP;

//go through the chain add contruct the boundary

do{

chainCodeFile >> chainDir;

loadNeighborsCoord(currentP);

nextP = neighborhood[chainDir];

imgAry[nextP.row][nextP.col] = tmplabel;

currentP = nextP;

}while(currentP != startP);

}//constructBoundary

void fillInterior(int \*\*imgAry, int numRows, int numCols, int tmpLabel){

//row by row, LR->TB increment all 0 pixels in btwn boundry pixels

for(int i = 0; i < numRows; i++){

int indexFirstB = -1;

int indexLastB = -1;

int numBoundOnLine = 0;

for(int j = 0; j < numCols; j++){

if(imgAry[i][j] == tmpLabel){

numBoundOnLine++;

if(indexFirstB == -1) indexFirstB = j;

else if( j > indexLastB) indexLastB = j;

continue;

}//if boundary px

}//for cols

for(int j = 0; j < numCols; j++){

if(j > indexFirstB && j < indexLastB) imgAry[i][j]++;

}

}//for rows

printImageAryToConsole(imgAry, numRows, numCols);

//col by col, TB->LR increment all 0 pixels in btwn boundry pixels

for(int j = 0; j < numCols; j++){

int indexFirstB = -1;

int indexLastB = -1;

for(int i = 0; i < numRows; i++){

if(imgAry[i][j] >= tmpLabel){

if(indexFirstB == -1) indexFirstB = i;

else if( i > indexLastB) indexLastB = i;

continue;

}//if boundary px

}//for cols

for(int i = 0; i < numRows; i++){

if(i > indexFirstB && i < indexLastB) imgAry[i][j]++;

}

}//for rows

printImageAryToConsole(imgAry, numRows, numCols);

//if a px was icremented twice, then it is part of the interior

for(int i = 0; i < numRows; i++){

bool aok = false;

for(int j = 0; j < numCols; j++){

if(imgAry[i][j] >= 3){

if(j+1 < numCols){

if(imgAry[i][j+1] < 3){

if(!aok) aok = true;

else aok = false;

}

else aok = false;

}

}

else if(imgAry[i][j] == 2){

if(aok){

if(j-1 > 0){

if(imgAry[i][j-1] < 2){

imgAry[i][j] = 0;

aok = false;

}

}

if(j+1 < numCols){

if(imgAry[i][j+1] < 2){

imgAry[i][j] = 0;

aok = false;

}

}

if(i+1 < numCols){

if(imgAry[i+1][j] < 2){

imgAry[i][j] = 0;

aok = false;

}

}

if(i-1 > 0){

if(imgAry[i-1][j] < 2){

imgAry[i][j] = 0;

aok = false;

}

}

//all good?

if(imgAry[i][j] == 2)

imgAry[i][j] = tmpLabel;

}

else imgAry[i][j] = 0;

}

else{

imgAry[i][j] = 0;

aok = false;

}

}//for cols

}//for rows

printImageAryToConsole(imgAry, numRows, numCols);

for(int i = 0; i < numRows; i++){

for(int j = 0; j < numCols; j++){

if(imgAry[i][j] > 2) imgAry[i][j] = tmpLabel - 2;

}

}

printImageAryToConsole(imgAry, numRows, numCols);

}//fillInterior

void printImageAryToFile(int \*\*&imgAry, int numRows, int numCols, ofstream &outputFileStream){

for(int i = 0; i < numRows; i++){

for(int j = 0; j < numCols; j++){

outputFileStream << imgAry[i][j] << " ";

}//for

outputFileStream << endl;

}//for

}//printImageToFile

void printImageAryToConsole(int \*\*&imgAry, int numRows, int numCols){

for(int i = 0; i < numRows; i++){

for(int j = 0; j < numCols; j++){

cout << imgAry[i][j] << " ";

}//for

cout << endl;

}//for

cout << endl;

}//printImageToFile

//OPERATORS

ChainCode operator=(const ChainCode& c){

ChainCode d;

return d;

}

};//ChainCode Class

/\* Class Constructors \*/

Image::Image(){

}//Image::Image()

Image::Image(ifstream &imgFileInputStream){

imgFileInputStream >> numRows;

imgFileInputStream >> numCols;

imgFileInputStream >> minVal;

imgFileInputStream >> maxVal;

setZeroFrame(zeroFramedAry);

loadImage(imgFileInputStream, zeroFramedAry);

int\*\* ary;

// ary = new int\*[numRows];

// for(int i = 0; i < numRows; i++){

// for(int j = 0; j < numCols; j++){

// ary[i][j] = 0;

// }//for

// }//for

imgAry = ary;

}//Image::Image(...)

/\* Data \*/

ifstream inFile;

ofstream chainCodeFile, decompressFile, boundaryFile;

Image image;

ChainCode chaincode;

/\* Function Headers \*/

/\* Program Begin \*/

int main(int argc, char \*argv[]){

//Step -1

if(argc != 2){

cout << "Correct syntax is >> ./chainCode <inputFile>\n where inputFile is a binary image containgina single object and a header.\n";

return -1;

}

string f = argv[1];

string mainFileName = f.substr(0,f.find(".txt",0));

//Step 0 (encomppases step 3)

inFile.open(argv[1]);

image = Image(inFile);

chaincode = ChainCode();

//Step 1

string chainCodeFileName = mainFileName + "\_chaincode.txt";

char ccCharAry[chainCodeFileName.length() + 1];

strcpy(ccCharAry, chainCodeFileName.c\_str());

//Step 2

chainCodeFile.open(ccCharAry);

//Step 3

chaincode.getChainCode(image, chainCodeFile);

//Step

chainCodeFile.close();

//Step

ifstream inChaincodeFile;

inChaincodeFile.open(ccCharAry);

//Step 4

string decompressedFileName = mainFileName + "\_chainCodeDecompressed.txt";

string boundaryFileName = mainFileName + "\_Boundary.txt";

char deCharAry[decompressedFileName.length() + 1];

char boundCharAry[boundaryFileName.length() + 1];

strcpy(deCharAry, decompressedFileName.c\_str());

strcpy(boundCharAry, boundaryFileName.c\_str());

//Step 5

decompressFile.open(deCharAry);

boundaryFile.open(boundCharAry);

//Step 6

chaincode.reconstructObject(inChaincodeFile, boundaryFile, decompressFile, image.imgAry);

//Step 7

inFile.close();

chainCodeFile.close();

decompressFile.close();

boundaryFile.close();

return 0;

}//main

**Input File**

20 40 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 1 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 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 0 0 0 0 0 0 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 1 1 1 1 1 0 0 0 0 0 0 0

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

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

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

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

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

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

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

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

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

**Chaincode File**

20 40 0 1

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

**Boundary File**

20 40 0 3

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 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 3 0 0 0 0 0 0 0 0

0 0 0 0 0 3 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 3 0 0 0 0 0 0 0

0 0 0 0 3 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 3 0 0 0 0 0 0

0 0 0 0 0 3 3 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 3 0 0 0 0 0

0 0 0 0 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0

0 0 0 0 3 3 3 3 0 3 0 0 0 0 0 0 3 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0

0 0 0 0 3 0 0 0 0 3 0 0 0 0 0 3 0 3 0 0 0 0 0 3 3 3 0 0 0 0 0 0 0 0 0 3 3 3 0 0

0 0 0 0 3 0 0 0 0 3 0 0 0 0 3 0 0 0 3 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 3 0 0 0 0 0

0 0 0 0 3 0 0 0 0 3 0 0 0 3 0 0 0 0 3 0 0 0 0 0 0 0 0 3 3 0 0 3 3 3 0 0 0 0 0 0

0 0 0 0 3 0 0 0 0 0 3 0 3 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 3 0 0 0 0 0 0 0 0

0 0 0 0 3 0 0 0 0 0 3 3 0 0 0 0 0 0 0 0 3 0 0 3 3 3 3 3 3 3 0 0 3 3 3 3 3 3 0 0

0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0

0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 3 0 0 0 0 0 0 0 0 3 3 0 0 0 0

0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 3 3 0 3 3 0 0 0 0 3 0 0 0 0 0 0

0 0 0 0 0 0 3 0 0 0 0 0 3 3 3 0 0 0 0 0 0 3 3 3 0 0 3 0 3 0 0 0 3 0 0 0 0 0 0 0

0 0 0 0 0 0 0 3 0 0 0 3 0 0 0 3 3 0 0 0 3 3 3 0 0 0 3 0 0 3 0 3 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 3 0 3 0 0 0 0 0 0 3 0 3 0 0 0 3 0 3 0 0 0 0 3 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 3 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**Decompressed File**

20 40 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 1 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 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 0 0 0 0 0 0 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 1 1 1 1 1 0 0 0 0 0 0 0

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

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

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

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

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

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

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

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

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