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

CSCI 381-26 Project 3: 3 noiseFilters Language: C++

Name: Akshar Patel

Due date: Soft copy: 2/18/2020

Hard copy: 2/20/2020

Submission date: Soft copy: 2/18/2020

Hard copy: 2/20/2020

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

**Part 1: Algorithm**

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

I. Main(...)

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

step 0: inFile 🡨 open

maskFile, AvgOutImg, MedianOutImg, GaussOutImg, DebugFile 🡨 open

step 1: numRows, numCols, minVal, maxVal 🡨 read from inFile

maskRows, maskCols, maskMin, maskMax 🡨 read from maskFile

step 2: dynamically allocate all 1-D and 2-D arrays

step 3: loadMask (maskAry, mask) // load mask onto maskAry

step 4: loadImage (mirrorFramedAry, inFile)

step 5: mirrorFraming (mirrorFramedAry)

step 6: ComputeAvgImg (avgAry, newMin, newMax) // see algorithm below

step 7: outputAryToFile(avgAry, AvgOutImg, newMin, newMax)

step 8: computeMedianImg (medianAry, newMin, newMax) // see algorithm

below

step 9: outputAryToFile(medianAry, MedianOutImg, newMin, newMax)

step 10: computeGaussImg (GaussAry, newMin, newMax) // see algorithm

below

step 11: outputAryToFile(GaussAry, GaussOutImg, newMin, newMax)

step 12: close all files

**Part 2: Source code**

#include <iostream>

#include<fstream>

using namespace std;

class imageProcessing {

public:

int numRows, numCols, minVal, maxVal, maskRows, maskCols, maskMin, maskMax;

int\*\* mirrorFramedAry;

int\*\* avgAry;

int\*\* medianAry;

int\*\* GaussAry;

int\*\* maskAry;

void set2DZero(int\*\* Ary){

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

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

Ary[i][j]=0;

}

}

}

void set2DMaskZero(int\*\* Ary){

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

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

Ary[i][j]=0;

}

}

}

void loadMask(int\*\* Ary, ifstream& mask){

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

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

mask>>Ary[i][j];

}

}

}

void loadImage(int\*\* Ary, ifstream& file){

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

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

file>>Ary[i][j];

}

}

}

void mirrorFraming(int\*\* Ary){

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

Ary[i][0]=Ary[i][1];

Ary[i][numCols+1]=Ary[i][numCols];

}

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

Ary[0][j]=Ary[1][j];

Ary[numRows+1][j]=Ary[numRows][j];

}

}

void ComputeAvgImg(int\*\* Ary, int \*newMin, int \*newMax){

\*newMin = 9999;

\*newMax = 0;

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

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

avgAry[i][j] = avg3by3(i,j,avgAry);

if(\*newMin > avgAry[i][j]){

\*newMin = avgAry[i][j];

}

if(\*newMax < avgAry[i][j]){

\*newMax = avgAry[i][j];

}

}

}

}

int avg3by3(int i, int j, int\*\* Ary){

int val;

int sum=0;

for(int x=i-1; x<i+2; x++){

for(int y=j-1; y<j+2 ; y++){

sum += mirrorFramedAry[x][y];

}

}

val = sum/9;

return val;

}

void outputAryToFile(int\*\* Ary, fstream& outfile, int x, int y){

outfile<<numRows<<" "<<numCols<<" "<<x<<" "<<y<<endl;

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

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

outfile<<Ary[i][j]<<" ";

}

outfile<<endl;

}

}

void computeMedianImg(int\*\* medianAry, int \*newMin, int \*newMax){

int neighborAry[9];

\*newMin = 9999;

\*newMax = 0;

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

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

loadNeighbors(i, j, neighborAry);

sort(neighborAry);

medianAry[i][j] = neighborAry[4];

if(\*newMin > medianAry[i][j]){

\*newMin = medianAry[i][j];

}

if(\*newMax < medianAry[i][j]){

\*newMax = medianAry[i][j];

}

}

}

}

void loadNeighbors(int i, int j, int neighborAry[]){

int z=0;

for(int x=i-1; x<i+2; x++){

for(int y=j-1; y<j+2 ; y++){

neighborAry[z] =mirrorFramedAry[x][y];

z++;

}

}

}

void sort(int neighborAry[]){

for(int i = 0; i < 9 ; i++){

for(int j = 0; j < 8; j++){

if(neighborAry[j] > neighborAry[j+1]){

swap(neighborAry[j], neighborAry[j+1]);

}

}

}

}

void computeGaussImg(int\*\* GaussAry, int \*newMin, int \*newMax){

\*newMin = 9999;

\*newMax = 0;

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

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

GaussAry[i][j] = convolution(i, j, GaussAry, maskAry);

if(\*newMin > GaussAry[i][j]){

\*newMin = GaussAry[i][j];

}

if(\*newMax < GaussAry[i][j]){

\*newMax = GaussAry[i][j];

}

}

}

}

int convolution(int i, int j, int\*\* GaussAry,int\*\* maskAry){

int sum=0, val;

sum=(mirrorFramedAry[i-1][j-1]\*maskAry[0][0])+

(mirrorFramedAry[i-1][j] \* maskAry[0][1])+

(mirrorFramedAry[i-1][j+1] \* maskAry[0][2])+

(mirrorFramedAry[i][j-1] \* maskAry[1][0])+

(mirrorFramedAry[i][j] \* maskAry[1][1])+

(mirrorFramedAry[i][j+1] \* maskAry[1][2])+

(mirrorFramedAry[i+1][j-1] \* maskAry[2][0])+

(mirrorFramedAry[i+1][j] \* maskAry[2][1])+

(mirrorFramedAry[i+1][j+1] \* maskAry[2][2]);

val = sum/22;

return val;

}

};

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

imageProcessing img;

int newMin,newMax;

ifstream inFile;

inFile.open(argv[1]);

ifstream maskFile;

maskFile.open(argv[2]);

fstream AvgOutImg;

AvgOutImg.open(argv[3], fstream::out);

fstream MedianOutImg;

MedianOutImg.open(argv[4], fstream::out);

fstream GaussOutImg;

GaussOutImg.open(argv[5], fstream::out);

inFile>>img.numRows;

inFile>>img.numCols;

inFile>>img.minVal;

inFile>>img.maxVal;

maskFile>>img.maskRows;

maskFile>>img.maskCols;

maskFile>>img.maskMin;

maskFile>>img.maskMax;

img.mirrorFramedAry = new int\* [img.numRows + 2];

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

img.mirrorFramedAry[i] = new int[img.numCols+2];

}

img.avgAry = new int\* [img.numRows + 2];

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

img.avgAry[i] = new int[img.numCols+2];

}

img.medianAry = new int\* [img.numRows + 2];

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

img.medianAry[i] = new int[img.numCols+2];

}

img.GaussAry = new int\* [img.numRows + 2];

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

img.GaussAry[i] = new int[img.numCols+2];

}

img.maskAry = new int\* [img.maskRows];

for(int i=0; i<img.maskRows; i++){

img.maskAry[i] = new int[img.maskCols];

}

img.set2DZero(img.mirrorFramedAry);

img.set2DZero(img.avgAry);

img.set2DZero(img.medianAry);

img.set2DZero(img.GaussAry);

img.set2DMaskZero(img.maskAry);

img.loadMask(img.maskAry, maskFile);

img.loadImage(img.mirrorFramedAry, inFile);

img.mirrorFraming(img.mirrorFramedAry);

img.ComputeAvgImg(img.avgAry, &newMin, &newMax);

img.outputAryToFile(img.avgAry, AvgOutImg, newMin, newMax);

img.computeMedianImg(img.medianAry, &newMin, &newMax);

img.outputAryToFile(img.medianAry, MedianOutImg, newMin, newMax);

img.computeGaussImg(img.GaussAry, &newMin, &newMax);

img.outputAryToFile(img.GaussAry, GaussOutImg, newMin, newMax);

inFile.close();

maskFile.close();

AvgOutImg.close();

MedianOutImg.close();

GaussOutImg.close();

}

**Part 3: Output**

|  |  |  |
| --- | --- | --- |
| **histogram of AvgOutImg** | **histogram of MedianOutImg** | **histogram of GaussOutImg** |
| 46 46 1 54  0 0  1 8  2 253  3 321  4 262  5 66  6 36  7 102  8 102  9 63  10 20  11 22  12 50  13 58  14 41  15 14  16 20  17 36  18 32  19 10  20 11  21 13  22 13  23 12  24 16  25 13  26 7  27 15  28 16  29 17  30 9  31 15  32 28  33 24  34 18  35 9  36 10  37 24  38 27  39 26  40 20  41 30  42 34  43 52  44 46  45 18  46 20  47 17  48 15  49 10  50 5  51 5  52 2  53 1  54 3 | 46 46 1 58  0 0  1 9  2 494  3 299  4 494  5 214  6 4  7 4  8 31  9 3  10 0  11 2  12 1  13 2  14 0  15 0  16 1  17 0  18 3  19 0  20 2  21 0  22 4  23 0  24 9  25 4  26 0  27 1  28 24  29 2  30 6  31 4  32 6  33 4  34 18  35 2  36 0  37 1  38 18  39 1  40 6  41 17  42 9  43 10  44 3  45 7  46 0  47 2  48 392  49 0  50 0  51 0  52 0  53 0  54 0  55 0  56 0  57 0  58 4 | 46 46 1 57  0 0  1 12  2 281  3 330  4 294  5 96  6 61  7 39  8 44  9 48  10 65  11 36  12 32  13 23  14 19  15 19  16 28  17 32  18 25  19 22  20 19  21 13  22 15  23 9  24 15  25 8  26 10  27 9  28 12  29 17  30 16  31 9  32 21  33 29  34 15  35 22  36 16  37 23  38 20  39 18  40 19  41 31  42 38  43 28  44 37  45 36  46 29  47 28  48 17  49 8  50 10  51 6  52 1  53 2  54 2  55 2  56 0  57 1 |

PrettyPrint of threshold AvgOutImg at 32

1 1

1 1 1

1

1 1 1 1

1 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1

1 1

1

PrettyPrint of threshold MedianOutImg at 32

1 1

1 1 1 1

1 1 1 1 1

1 1 1

1 1 1 1 1 1 1 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1

1 1

PrettyPrint of threshold GaussOutImg at 32

1 1 1

1

1 1 1

1 1 1 1

1 1 1

1 1

1 1 1 1 1 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1 1

1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1

1 1 1

1

**For extra credit**

* running histogram on project 2

The selected threshold value for AvgOutImg= 31

The selected threshold value for MedianOutImg= 31

The selected threshold value for GaussOutImg= 31

* running project 1 using selected threshold value

PrettyPrint of threshold AvgOutImg at 31

1 1 1

1 1 1 1

1

1 1 1 1

1 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1

1

PrettyPrint of threshold MedianOutImg at 45

1 1

1 1 1 1

1 1

1 1

1 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1

1

1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1

1

PrettyPrint of threshold GaussOutImg at 31

1 1 1 1

1

1 1 1

1 1 1 1 1

1 1 1 1 1

1 1

1 1 1 1 1 1 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

1 1 1

1