CSC 381-34: Proj2 (C++)

Swrajit Paul

Due date: Sept. 14, 2018

Algorithm steps:

```
III. Main( )
*****************************

step 0: - open the input file and output file
        - read the image header, the four numbers
        - dynamically allocate all 1-D and 2-D array

        - thr_value ← read from argv[1]

>***<       - whichMethod ← ask user from console for the method:

           // inform user that 1 is for averaging,
           // and 2 is for median
           // if user types other number, ask the user
           // again


step 1: loadImage (imgInAry, mirrorFramedAry)
     // read from input file and load onto imgInAry begins at [0][0]
           // and load onto mirrorFramedAry begin at [1,1]

step 2: ComputeHistogram(imgInAry, hist, maxVal)

step 3: printHist(hist) // see the format in the above.

step 4: mirrowFramed (mirrorFramedAry) // Use the algorithm given in
class

>****< Modify step 5

step 5: if whichMethod == 1

           computeAVG3X3 (mirrorFramedAry,  tempAry)

     else if  whichMethod == 2

           computeMEDIAN3X3 (mirrorFramedAry,  tempAry)
                // see algorithm below

     else write error method and exit the program!


Step 6: computThreshold (tempAry,imgOutAry)

Step 7: prettyPrint (imgOutAry)
```

```
step 8: output the image header (numRows, numCols, newMin, newMax)
     to Output2(argv[3]): the result of thresholded  image

step 9: output tempAry, begin at [1,1], within the frame, to
Output2(argv[3])

step 10: close all files




************************************
IV. computeMEDIAN3X3 (mirrorFramedAry,  tempAry)
************************************

step 1: process the MirrorframedAry, from left to right and top to
bottom
       using i, and j, begin at (1, 1) until one before the last row.

      p(i,j) <-- next pixel

Step 2: loadNeighbors (neighborAry)
     // load the 3 x 3 neighbors of p(i,j)into neightAry

step 3: sort(neighborAry)

        tempAry(i,j) <-- neighborAry[4]

       - keep tracking the newMin and newMax of tempAry

step 4: repeat step 1 - step3  until all pixels inside of the framed
are processed
```

PLUS FROM PROJECT ONE!

```
III. Main( )
*****************************

step 0: - open the input file and output file
       - read the image header, the four numbers
       - dynamically allocate all 1-D and 2-D array

       - thr_value <-- read from argv[1]

step 1: loadImage (imgInAry, mirrorFramedAry)
          // read from input file and load onto imgInAry begins at [0][0]
          // and load onto mirrorFramedAry begin at [1,1]

step 2: ComputeHistogram(imgInAry, hist, maxVal)

step 3: printHist(hist) // see the format in the above.
```

```
step 4: mirrowFramed (mirrorFramedAry) // Use the algorithm given in class

step 5: computeAVG3X3 (mirrorFramedAry,  tempAry) // see algorithm below

Step 6: computThreshold (tempAry,imgOutAry)

Step 7: prettyPrint (imgOutAry)

step 8: output the image header (numRows, numCols, newMin, newMax)
     to Output2(argv[3]): the result of thresholded  image

step 9: output tempAry, begin at [1,1], within the frame, to Output2(argv[3])

step 10: close all files

*******************************
VI. computeHistogram(imgInAry, maxVal, hist )
*******************************

step 1: dynamically allocate the hist array size maxVal+1 and initialize to 0

step 2: // process imgInAryfrom left to right and top to bottom

     p(i,j) <- next pixel
     hist[p(i,j)]++

step 3: repeat step 2 until all pixels are processed.


************************************
IV. computeAVG3X3 (mirrorFramedAry,  tempAry)
************************************

step 1: process the MirrorframedAry, from left to right and top to bottom
       using i, and j, begin at (1, 1) until one before the last row.

      p(i,j) <-- next pixel

Step 2: loadNeighbors (neighborAry)
     // load the 3 x 3 neighbors of p(i,j)into neightAry

step 3: tempAry(i,j) <-- Avg3x3(neighborAry)
          // compute the averaging of neighborAry

       - keep tracking the newMin and newMax of tempAry

step 4: repeat step 1 - step3  until all pixels inside of the framed are
processed


*******************************
VI. computeThreshold(MirrorframedAry, imgOutAry, thr_value)
*******************************

step 1: process the MirrorframedAry, from left to right and top to bottom
```

```
        using i, and j, begin at (1, 1) until one before the last row.

       p(i,j) <-- next pixel

      if (p(i,j)>= thr_value
            imgOutAry(i,j) <--  1
      else
            imgOutAry(i,j) <--  0
```

step 2: repeat step 1 until all pixels are processed.

```
************************************
III. prettyPrint (imgOutAry)
************************************
```

step 1: outFile <-- open Output2(argv[3]

step 2: // process imgOutAry from left to right and top to bottom

```
      p(i,j) <- next pixel
      if p(i,j) > 0
         output p(i,j) to outFile
     else
         output '  ' // 2 blanks to outFile
```

step 3: repeat step 2 until all pixels are processed.

SOURCE CODE

```
#include <iostream>
#include <fstream>
#include <sstream>

using namespace std;

ifstream inFile;
ofstream outFile;
ofstream outFiletwo;
ofstream outFilethree;

class imageProcessing {

	public:

		int numRows;
		int numCols;
		int minVal;
		int maxVal;
		int newMin;
		int newMax;

		int thr_value;
		int** imgInAry;
		int** imgOutAry;
		int** mirrorFramedAry;
		int** tempAry;
	   int* hist;
	   int neighborAry[9];

	   imageProcessing(string in, string intwo, string out, string outtwo, string outthree) {

		inFile.open(in.c_str());

			stringstream s(intwo);
			s >> thr_value;
			s.clear();
			outFile.open(out.c_str());

			outFiletwo.open(outtwo.c_str());

			outFilethree.open(outthree.c_str());

			inFile >> numRows;
			inFile >> numCols;
			inFile >> minVal;
			inFile >> maxVal;


			imgInAry = new int*[numRows];
			for(int i = 0; i < numRows; i++){
				imgInAry[i] = new int[numCols];
			} // set up the array with proper rows and cols
			for(int i = 0; i < numRows; i++) {
```

```
		for(int j = 0; j < numCols; j++) {
			imgInAry[i][j] = 0;
		}
	}// initialize the array

	imgOutAry = new int*[numRows];
	for(int i = 0; i < numRows; i++){
		imgOutAry[i] = new int[numCols];
	}// set up the array with proper rows and cols
	for(int i = 0; i < numRows; i++) {
		for(int j = 0; j < numCols; j++) {
			imgOutAry[i][j] = 0;
		}
	}// initialize the array

	mirrorFramedAry = new int*[numRows+2];
	for(int i = 0; i < numRows+2; i++){
		mirrorFramedAry[i] = new int[numCols+2];
	}// set up the array with proper rows and cols
	for(int i = 0; i < numRows+2; i++) {
		for(int j = 0; j < numCols+2; j++) {
			mirrorFramedAry[i][j] = 0;
		}
	}// initialize the array

	tempAry = new int*[numRows+2];
	for(int i = 0; i < numRows+2; i++){
		tempAry[i] = new int[numCols+2];
	}// set up the array with proper rows and cols
	for(int i = 0; i < numRows+2; i++) {
		for(int j = 0; j < numCols+2; j++) {
			tempAry[i][j] = 0;
		}
	}// initialize the array

	hist = new int[maxVal+1];
	for(int j = 0; j < maxVal+1; j++) {
		hist[j] =0;
	}


}

void loadImage(int** imgInAry, int** mirrorFramedAry) {

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

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

			inFile >> imgInAry[i][j];

		}
	}
```

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

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

			mirrorFramedAry[i+1][j+1] = imgInAry[i][j];

		}

	}

}

void ComputeHistogram(int** imgInAry, int* hist, int mVal) {

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

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

			hist[imgInAry[i][j]] += 1;

		}
	}



}

void printHist(int* hist) {

	outFile << numRows << " " << numCols << " " << minVal << " " << maxVal << endl;

	for(int j = 0; j < maxVal+1; j++) {

		outFile << j << " " << hist[j] << endl;

	}

	outFile.close();
}

void mirrowFramed (int** mirrorFramedAry) {

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

		mirrorFramedAry[0][j] = 0;
		mirrorFramedAry[numRows+1][j] = 0;

	}

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

		mirrorFramedAry[j][0] = 0;
		mirrorFramedAry[j][numCols+1] = 0;

	}
```

```
}

void computeAVG3X3 (int** mirrorFramedAryw,int** tempAryw) {

	newMin = 20000000;
	newMax = 0;

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

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

			neighborAry[0] = mirrorFramedAryw[i-1][j-1];
			neighborAry[1] = mirrorFramedAryw[i-1][j];
			neighborAry[2] = mirrorFramedAryw[i-1][j+1];
			neighborAry[3] = mirrorFramedAryw[i][j-1];
			neighborAry[4] = mirrorFramedAryw[i][j];
			neighborAry[5] = mirrorFramedAryw[i][j+1];
			neighborAry[6] = mirrorFramedAryw[i+1][j-1];
			neighborAry[7] = mirrorFramedAryw[i+1][j];
			neighborAry[8] = mirrorFramedAryw[i+1][j+1];



			int sum =0;

			for (int k =0; k < 9; k++) {
				sum += neighborAry[k];
			}

			int avg = sum / 9;

			tempAryw[i][j] = avg;

			if (avg <= newMin) {
				newMin = avg;
			}
			if (avg >= newMax) {
				newMax = avg;
			}

		}

	}

}

void computeMEDIAN3X3 (int** mirrorFramedAryw,int** tempAryw) {

	newMin = 20000000;
	newMax = 0;

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

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

			neighborAry[0] = mirrorFramedAryw[i-1][j-1];
			neighborAry[1] = mirrorFramedAryw[i-1][j];
			neighborAry[2] = mirrorFramedAryw[i-1][j+1];
			neighborAry[3] = mirrorFramedAryw[i][j-1];
			neighborAry[4] = mirrorFramedAryw[i][j];
			neighborAry[5] = mirrorFramedAryw[i][j+1];
			neighborAry[6] = mirrorFramedAryw[i+1][j-1];
			neighborAry[7] = mirrorFramedAryw[i+1][j];
			neighborAry[8] = mirrorFramedAryw[i+1][j+1];

			sort(neighborAry);
			tempAryw[i][j] = neighborAry[4];


			if (neighborAry[4] <= newMin) {
				newMin = neighborAry[4];
			}
			if (neighborAry[4] >= newMax) {
				newMax = neighborAry[4];
			}

		}

	}

}

void sort(int array[]){
	int i, j;
	for (i =0; i < 9; i++){
		for(j =0; j < 9-i-1; j++){
			if(array[j] > array[j+1]){
				int temp = array[j];
				array[j] = array[j+1];
				array[j+1] = temp;
			}
		}
	}
}

void computeThreshold (int** tempAry, int** imgOutAry, int thr_value) {


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

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

			int pixel = tempAry[i][j];

			if(pixel >= thr_value) {
				imgOutAry[i-1][j-1] = 1;
			}
			else {
```

```
                                        imgOutAry[i-1][j-1]= 0;
                                }

                        }
                }

        }

        void prettyPrint (int** imgOutAryw) {

                outFiletwo << numRows << " " << numCols << " " << minVal << " " << maxVal << endl;

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

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

                                if(imgInAry[i][j] > 0) {
                                        outFiletwo << imgOutAryw[i][j] << " " ;
                                }
                                else {
                                        outFiletwo << " ";
                                }
                        }
                        outFiletwo << endl;
                }
        }


};

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

        imageProcessing img (argv[1],argv[2],argv[3],argv[4],argv[5]);

        int whichMethod = 0;
        while(true){
                cout << "which method would you like to use?" << endl;
                cout << "enter 1 for average filter or enter 2 for median filter" << endl;
                cin >> whichMethod;

                if (whichMethod == 1 || whichMethod == 2){
                        break;
                }
                else{
                        cout << "please re-enter!" << endl;
                }
        }

        img.loadImage(img.imgInAry, img.mirrorFramedAry);

        img.ComputeHistogram(img.imgInAry, img.hist, img.maxVal);

        img.printHist(img.hist);

        img.mirrowFramed(img.mirrorFramedAry);
```

```
	if(whichMethod == 1){
		img.computeAVG3X3(img.mirrorFramedAry, img.tempAry);
	}
	else if(whichMethod == 2){
		img.computeMEDIAN3X3(img.mirrorFramedAry, img.tempAry);
	}
	else if(whichMethod >= 3 || whichMethod <= 0){
		return 0;
	}

	img.computeThreshold (img.tempAry, img.imgOutAry, img.thr_value);

	img.prettyPrint (img.imgOutAry);

	outFilethree << img.numRows << " " << img.numCols << " " << img.minVal << " " << img.maxVal << endl;

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

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

			if(img.tempAry[i][j] > 0) {
				outFilethree << img.tempAry[i][j] << " ";
			}
			else {
				outFilethree << " ";
			}
		}
		outFilethree << endl;
	}

	return 0;
}
```

INPUT

```
46 46 1 63
1 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
2 1 2 3 4 55 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
3 1 2 3 4 5 1 2 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 43 4 5 1 2 3 4 5 11 2 3 4 5 1 2 3 4
5
4 1 2 3 4 5 1 2 3 4 55 51 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4
5
5 1 22 3 4 5 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 38 4 5 1 2 43 4 35 1 2 3 4 5 1 2 3 4 5 1 2 3 4
5
6 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 41 4 5 1 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5 8 2 3 4 5 1 2 38 4 5 1 12 3 44 5 1 2 3 4 5 1 2 31 4 5 1 2 3 4
5
8 1 2 3 4 5 1 2 53 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 45 1 2 3 4 55 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 38 48 5 1 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4
5
10 11 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 4 48 48 48 1 2 43 4 5 1 2 3 4 5 1 2 3 4 5 1 2
3 4 5
1 1 2 3 4 5 11 2 3 4 5 1 2 3 4 5 1 2 3 4 48 33 4 4 41 48 48 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 3
4 5
2 1 2 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 3 48 48 48 4 8 48 48 48 48 3 4 5 19 2 3 4 5 1 2 3 14 5 1
2 3 4 5
3 1 2 3 4 55 51 2 3 4 5 1 2 3 4 5 1 2 4 8 4 8 8 4 4 4 8 8 8 4 5 1 2 3 4 5 1 2 3 44 5 1 2 3 4 5
4 21 22 23 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28
24 22 20 18 18 16 13 4 5 1 2 3 14 5
5 1 21 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 48 48 10 48 48 48 34 48 48 48 48 5 1 2 3 4 5 1 2 3 4
5 1 32 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 48 48 48 48 48 48 48 48 48 48 48 4 48 48 1 2 3 4 5 1 2 3 4
55 1 2 3 4 5
7 1 2 3 14 5 1 2 3 4 5 1 2 3 4 48 48 48 41 42 43 41 42 43 4 48 48 46 48 48 48 48 2 3 4 51 1 2
3 4 5 1 2 3 4 5
8 1 2 3 4 5 1 2 3 4 5 1 2 3 48 41 48 44 48 48 45 48 44 48 48 48 48 48 48 48 4 4 48 3 4 5 1 2 3
4 5 1 2 3 4 5
9 1 2 3 4 15 1 12 3 4 5 1 2 48 48 48 48 60 48 48 48 48 48 61 62 48 48 48 48 8 7 48 48 48 4 5 1
2 3 4 5 1 2 13 4 5
10 1 2 3 4 5 1 2 3 4 5 1 48 48 48 5 48 48 48 3 48 48 48 48 48 6 48 48 47 48 8 48 48 48 48 5 1
12 3 4 5 1 2 3 4 5
1 1 52 3 4 5 1 12 3 4 5 48 48 58 48 48 48 48 48 48 28 38 48 48 48 48 48 48 8 48 48 28 48 48 48
48 1 2 3 4 5 11 2 3 4 5
2 1 2 3 4 5 1 2 3 4 48 48 58 48 48 48 40 48 47 48 48 48 41 48 42 48 52 48 4 8 5 48 48 48 38 48
48 48 3 4 5 1 2 3 14 5
3 21 22 23 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28
24 22 20 18 18 16 13 4 5 1 2 3 14 5
4 1 2 3 4 5 1 2 48 48 48 48 48 48 4 48 48 48 48 58 58 58 38 38 58 48 58 58 28 24 14 48 48 48
48 48 48 48 48 4 5 1 2 3 4 5
5 1 2 48 41 48 42 48 43 8 48 60 48 48 48 48 41 42 48 43 48 46 48 45 48 40 48 4 3 48 30 48 48
48 8 48 48 48 38 48 48 28 48 48 4 5
6 1 2 3 4 5 1 2 48 48 48 48 48 48 48 48 48 63 48 63 48 48 48 48 4 48 48 48 8 4 48 48 48 48 48
18 48 48 48 4 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 48 48 8 48 48 42 48 48 18 48 48 48 48 63 48 48 48 48 48 48 8 8 48 48 48 5 48
48 48 3 4 5 1 2 3 4 5
8 1 2 3 4 5 13 2 3 4 48 48 62 48 55 48 48 48 4 7 8 48 48 48 54 48 58 48 48 4 4 8 48 48 48 48
48 2 3 4 5 11 2 3 4 5
9 1 2 3 4 5 1 2 3 4 5 48 48 48 48 48 48 48 28 8 48 48 48 48 48 48 18 48 48 6 4 8 4 48 48 48 1
2 3 4 5 1 2 3 4 5
10 1 2 3 4 5 1 2 3 4 5 1 48 48 48 48 3 48 48 48 48 48 48 18 48 48 48 48 48 48 48 8 4 48 48 5 1
2 3 4 5 1 12 3 4 5
1 21 22 23 24 27 28 29 31 30 32 34 35 34 35 38 40 48 60 63 60 48 41 38 35 34 32 31 30 28 25 28
24 22 20 18 18 16 13 4 5 1 2 3 14 5
```

```
2 1 2 3 4 5 1 2 3 4 5 1 2 48 48 48 48 48 18 48 48 48 48 48 48 48 48 48 8 48 48 8 4 48 4 5 1 2
3 14 15 1 2 3 4 5
3 11 2 3 4 5 1 2 3 4 5 1 2 3 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 4 3 4 5 1 2
3 4 5 1 2 3 4 5
4 1 2 3 4 15 1 2 3 4 5 1 2 3 4 48 48 41 42 43 48 40 48 42 48 43 48 44 48 28 48 48 2 3 4 5 1 2
3 4 55 1 2 3 4 5
5 1 2 3 42 55 1 42 3 4 5 1 2 3 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 58 4 1 8 4 1 48 2 4 8 48 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4
5
7 1 2 3 4 5 1 2 3 4 5 13 2 3 4 5 1 2 48 48 8 48 4 5 4 48 48 8 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3
4 5
8 1 2 3 4 51 1 2 3 4 5 1 2 3 4 5 1 2 3 48 38 48 38 8 1 48 38 48 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2
3 4 5
9 1 2 3 4 5 1 12 3 4 5 1 2 3 4 5 1 2 3 4 48 48 48 48 48 48 48 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2
3 4 5
10 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 18 48 48 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3
4 5
1 1 2 3 44 5 1 2 3 4 5 1 12 3 4 5 1 2 3 4 5 1 48 48 48 5 1 2 3 4 5 1 2 3 4 55 1 2 3 4 55 1 2 3
4 5
2 1 2 3 48 5 1 2 3 4 55 51 12 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4
5
3 1 2 3 4 45 51 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 43 5 1 2 3 4 5 1 2 3 4
5
4 1 2 3 4 5 1 2 3 4 5 1 2 3 14 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 39 54 5 1 2 43 4 5 1 2 33
4 5
5 1 2 3 4 5 11 2 3 44 5 1 2 3 4 5 1 2 3 4 5 1 2 48 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
 6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

OUTPUT

Histogram for data

```
46 46 1 63
0 0
1 277
2 278
3 270
4 319
5 278
6 7
7 6
8 35
9 4
10 5
11 7
12 8
13 6
14 9
15 3
16 3
17 0
18 12
19 1
20 3
21 4
22 7
23 3
24 7
25 3
26 0
27 3
```

28 15
29 3
30 7
31 7
32 7
33 2
34 10
35 10
36 0
37 0
38 17
39 1
40 6
41 12
42 10
43 15
44 12
45 6
46 2
47 2
48 363
49 0
50 0
51 8
52 2
53 1
54 2
55 11
56 0
57 0
58 10
59 0
60 8
61 1
62 2
63 6

AVG FILTER IMG

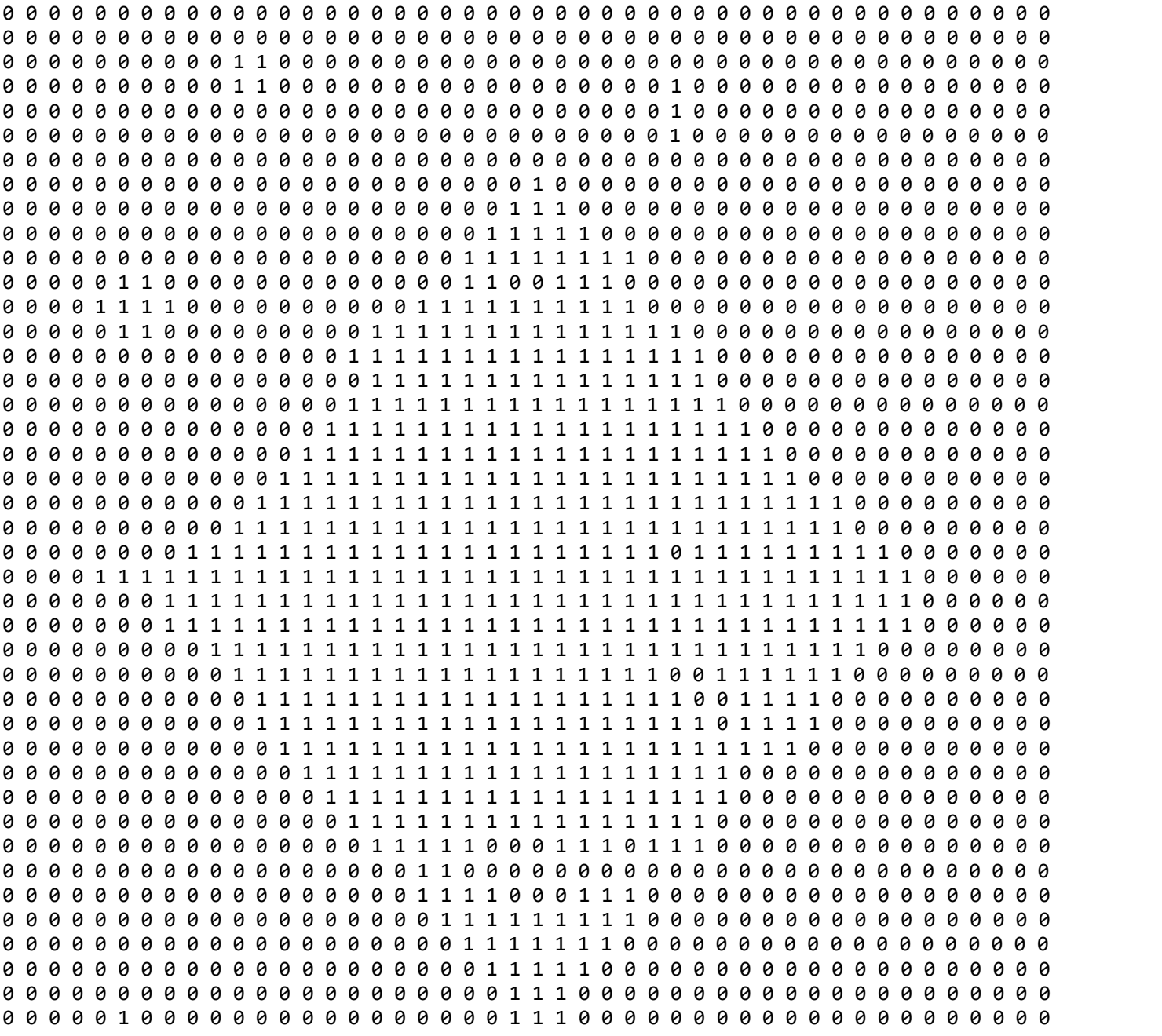
46 46 1 63

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

2 3 2 3 4 4 4 8 17 33 38 45 45 49 48 48 46 41 38 35 40 45 49 45 43 44 49 44 29 20 20 34 43 43 39 39 39 32 18 9 4 3 3 3 4
3 3 2 3 4 4 4 3 7 18 29 40 45 49 48 48 44 37 28 27 34 45 49 50 48 46 45 45 34 19 10 20 34 38 43 38 32 17 8 4 4 3 3 3 4
3 4 2 3 4 4 4 3 3 8 18 34 44 50 48 43 43 35 31 27 34 43 44 45 45 46 45 45 38 28 15 15 24 38 43 32 17 7 3 4 4 4 4 4 4
4 7 8 9 10 11 11 11 12 13 18 28 38 43 43 39 41 41 44 45 46 48 42 41 40 39 39 39 37 31 22 17 21 29 33 23 12 6 5 5 3 3 3 5 5
4 6 8 9 10 11 11 11 12 13 12 18 27 38 43 39 41 40 47 49 51 48 42 41 40 43 42 37 37 36 32 21 21 24 24 13 7 6 6 7 5 4 3 5 5
4 7 9 9 10 11 11 11 12 13 12 13 17 28 38 44 46 45 47 49 51 48 46 44 43 43 42 37 37 36 32 21 16 14 14 8 7 6 6 7 5 3 2 4 5
2 3 3 3 5 4 3 2 3 4 3 2 7 17 33 43 47 43 42 43 46 47 46 47 46 47 47 43 40 41 36 24 14 8 9 3 2 2 4 11 11 9 2 3 4
2 3 3 7 14 14 13 6 7 4 3 2 2 7 18 28 32 31 31 32 31 32 31 32 32 32 32 32 30 31 26 18 8 3 4 3 2 2 3 9 8 8 2 3 4
2 2 2 7 14 14 13 6 7 4 3 2 2 3 8 13 22 26 32 28 22 17 17 18 22 22 22 18 21 21 20 12 7 3 4 3 2 2 3 9 8 8 2 3 4
2 3 2 7 13 13 12 6 7 4 4 4 3 3 4 3 8 17 29 25 19 9 9 4 13 18 18 14 14 14 8 2 2 3 4 3 2 2 3 4 3 2 2 3 4
2 3 2 3 9 8 7 2 3 4 4 4 3 3 4 3 7 17 33 33 33 21 18 8 18 26 32 23 19 14 8 2 2 3 4 3 2 2 3 4 3 2 2 3 4
3 3 2 3 9 8 8 3 4 4 4 4 3 3 4 3 2 7 17 27 37 36 32 22 28 36 37 27 13 9 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4
3 4 2 3 9 8 8 3 4 4 3 2 2 3 4 3 2 2 7 17 32 41 39 33 35 36 31 16 8 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4
2 3 2 7 8 7 3 3 4 4 3 3 4 4 3 2 2 3 8 18 33 39 44 39 32 17 7 3 4 3 2 2 3 9 8 8 2 3 9 8 8 2 3 4
1 2 2 12 13 12 2 2 3 9 14 16 9 5 4 3 2 2 3 4 8 18 29 34 30 17 7 2 3 4 3 2 2 3 9 8 8 2 3 9 8 8 2 3 4
1 1 2 12 17 22 12 7 3 9 14 16 9 5 4 3 2 2 3 4 3 7 22 28 23 8 2 2 3 4 3 2 2 7 13 13 8 2 3 9 8 8 2 3 4
1 2 2 7 13 18 12 7 3 9 14 14 8 5 5 4 2 2 3 4 3 2 17 18 19 3 2 2 3 4 3 2 6 16 17 13 2 6 7 8 3 2 5 6 7
1 2 2 3 8 14 13 8 7 8 7 2 2 4 5 4 2 2 3 4 3 2 17 18 19 3 2 2 3 4 3 2 6 16 17 13 2 6 7 8 3 2 5 6 7
2 2 2 3 4 4 3 3 7 8 7 2 2 4 5 4 2 2 3 4 3 2 12 13 14 3 2 2 3 4 3 2 6 12 13 8 2 6 7 8 3 2 5 6 7

Threshold Value = 20

46 46 1 63



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

Threshold Value = 40

46 46 1 63
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 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 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 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 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 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 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 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 0 0 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 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 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 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 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 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 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 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 0 0 0 0 1 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 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 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 0 0 0 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 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 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 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 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 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 0 0 0 0 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 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 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 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 1 1 1 1 0 0 0 0 1 1 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 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 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 0 0 0 0 0 0 0 0 0 1 1 0 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 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 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 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 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 1 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 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 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 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 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 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 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0

MEDIAN FILTER IMG

46 46 1 63

1 1 2 3 1 1 1 2 3 1 1 1 2 3 1 1 1 2 3 1 1 1 2 3 1 1 1 2 4 1 1 1 2 3 1 1 1 2 3 1 1 1 2 3
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 5 5 3 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 5 5 3 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 5 4 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 5 4 38 38 5 2 2 3 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5 5 33 41 48 48 2 2 3 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 5 5 3 3 4 4 2 2 3 4 4 2 2 3 4 48 33 8 8 48 48 48 3 4 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 11 11 3 3 4 4 2 2 3 4 4 2 2 3 4 33 8 8 4 8 48 48 8 4 5 5 2 2 3 4 4 2 2 3 5 5 2 2 3 4
1 2 3 4 23 28 29 28 4 5 5 5 3 4 5 5 5 3 8 48 48 41 8 8 34 34 32 30 8 5 5 5 3 4 5 5 5 3 4 5 5 2 2 3 4
1 4 3 4 5 24 27 3 4 5 5 5 3 4 5 5 5 40 48 48 48 41 38 38 34 34 32 31 30 25 5 5 3 4 5 5 5 3 4 5 4 2 2 3 4
1 5 3 4 5 5 5 3 4 5 5 5 3 4 5 5 40 48 48 48 48 48 48 48 48 48 48 48 48 30 28 5 3 4 5 5 5 3 4 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5 48 48 48 48 48 48 48 48 48 48 48 48 48 48 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 48 48 48 48 48 48 45 48 48 48 48 48 48 48 48 48 4 3 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 48 48 48 48 48 48 48 45 48 48 48 48 48 48 48 48 48 48 4 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 47 8 48 48 48 5 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 5 3 4 4 4 5 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 47 48 48 48 48 48 5 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 5 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 48 48 48 48 48 48 12 3 4 4 4 2 2 3 4
1 2 3 4 5 5 5 3 4 5 32 48 48 48 48 48 48 48 48 48 48 48 48 42 48 48 48 32 30 25 28 28 48 48 48 38 18 13 4 4 4 2 2 3
5
1 2 3 4 5 5 5 3 29 32 48 48 48 48 48 40 48 48 48 58 58 48 41 41 42 48 48 32 28 24 25 28 48 48 48 48 48 48 13 5 4 2 2
3 5
1 3 3 22 24 27 28 31 31 43 48 48 48 48 48 41 48 48 48 58 58 48 45 41 40 48 40 31 28 28 28 30 48 48 48 48 48 48 38 13
5 5 3 4 5
1 2 2 3 5 5 5 42 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 24 24 48 48 48 48 48 48 48 48 48 38 5 5
3 4 4
1 2 2 3 5 5 5 3 43 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 8 48 48 48 48 48 48 48 48 48 5 5 5 3 4
4
1 2 2 3 4 4 2 2 3 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 8 48 48 48 48 48 48 48 4 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 48 48 48 48 48 48 48 48 28 28 48 48 48 48 48 48 48 48 48 8 8 8 48 48 48 48 48 3 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 5 48 48 48 48 48 48 48 48 28 48 48 48 48 48 48 48 48 48 48 8 8 8 48 48 48 2 2 3 4 4 5 3 3 4
1 2 3 4 5 5 5 3 4 5 5 34 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 30 25 8 22 24 48 18 5 3 4 4 4 2 2 3 4
1 2 3 4 5 5 5 3 4 5 5 5 34 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 28 24 22 22 20 5 5 3 4 5 4 2 2 3 4
1 2 3 4 5 5 5 3 4 5 5 5 3 35 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 28 24 8 4 5 5 5 3 4 5 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 8 4 4 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 48 48 42 42 43 43 48 42 48 43 48 44 48 44 48 28 4 3 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 5 41 42 42 4 4 4 4 4 4 4 4 8 8 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 4 4 48 8 4 4 4 4 4 4 4 4 4 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 48 48 48 8 8 4 5 38 48 8 8 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 38 48 48 48 8 48 48 48 38 4 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 48 48 48 48 48 48 48 3 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5 48 48 48 48 48 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 5 5 2 2 3 4 4 5 3 4 4 4 2 2 3 4 4 5 48 48 48 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 5 5 2 2 3 4 4 5 3 4 4 4 2 2 3 4 4 2 2 48 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4
1 2 2 3 4 5 2 2 3 4 4 5 3 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4
1 2 2 3 4 5 5 3 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4
1 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 3 5 4 2 2 3 4 4 2 2 3 4

Threshold Value = 20

46 46 1 63
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 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 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 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 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 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 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 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 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 0 0 1 1 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 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 1 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 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 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 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 1 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 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 1 1 1 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 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 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 0 0 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 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 1 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 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 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 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 1 1 1 1 1 1 1 0 0 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 1 1 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 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 1 1 1 1 1 1 0 0 0 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 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 1 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 1 1 1 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 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 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 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 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 1 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 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 1 1 1 0 0 0 0 1 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 1 1 1 1 0 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 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 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 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
0 0 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 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 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 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 0 0 0 0 0 0 0 0 0 0 0

Threshold Value = 40

```
46 46 1 63
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 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 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 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 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 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
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 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 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 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 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 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 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 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
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 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 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 0 0 0 1 1 1 1 1 1 1 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 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 0 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 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 1 1 1 1 1 1 1 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 1 1 1 1 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 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 1 1 1 1 1 1 1 0 0 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 1 1 1 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 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 0 0 1 1 1 1 1 1 1 1 1 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 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 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 0 0 1 1 1 1 1 1 1 1 1 1 1 1 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 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 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 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 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 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 0 0 0 0 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 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 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 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 1 1 1 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 1 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 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 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 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 0 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 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 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 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 0 0 0 0 0 0