Student: Pawan Bhatta

Project Due Date: 02/21/2021

Algorithm for Computing Average:

step 0: newMin ← 9999; newMax ← 0
step 1: r ← 1
step 2: c ← 1
step 3: avgAry [r,c] ← avg3x3 (r, c)
step 4: if newMin > avgAry [r,c]
newMin ←avgAry [r,c]
if newMax < avgAry [r,c]
newMax ← avgAry [r,c]

step 5: c++
step 6: repeat step 3 to step 5 while c < numCols+1
step 7: r++
step 8: repeat step 2 to step 7 while r < numRows+1

Algorithm for Computing Median:

step 0: newMin ← 9999; newMax ← 0
step 1: r ← 1
step 2: c ← 1
step 3: medianAry [r,c] ← median3x3 (r, c)
step 4: if newMin > medianAry [r,c]
newMin ← medianAry [r,c]
if newMax < medianAry [r,c]
newMax ← medianAry [r,j]

step 5: c++
step 6: repeat step 3 to step 5 while c < numCols+1
step 7: r++
step 8: repeat step 2 to step 7 while r < numRows+1

Algorithm for Computing Corner Preserving Filter:

step 0: newMin ← 9999; newMax ← 0
step 1: r ← 2
step 2: c ← 2
step 3: CPAry [r,c] ← CP5x5 (r, c)
step 4: if newMin > CPAry [r,c]
newMin ← CPAry [r,c]
if newMax < CPAry [r,c]
newMax ← CPAry [r,j]

step 5: c++
step 6: repeat step 3 to step 5 while c < numCols+2
step 7: r++
step 8: repeat step 2 to step 7 while r < numRows+2

Algorithm for Reformating Image:

step 0: newMin ← 9999; newMax ← 0
Step 1: OutImg ← output numRows, numCols, newMin, newMax
Step 2: str ← to_string(newMax) // a method in C++ string class
Width ← length of str
Step 3: r ← frameSize
Step 4: c ← frameSize
Step 5: OutImg ← inAry[r][c]
Step 6: str ← to_string (inAry[r][c])
WW ← length of str
Step 7: OutImg ← one blank space
WW ++
Step 8: repeat step 7 while WW < Width
Step 9: c++
Step 10: repeat Step 5 to Step 9 while c < (numCols + frameSize)
Step 11: r++
Step 12: repeat Step 4 to Step 10 while c < (numCols + frameSize)

Algorithm to create threshold image:

step 0: newMin ← 0
newMax ← 1
step 1: r ← frameSize
step 2: c ← frameSize
step 3: if ary1[r][c] >= thrVal
ary2[r][c] ← 1
else
ary2[r][c] ← 0

step 4: c++
step 5: repeat step 3 to step 4 while c < (numCols + frameSize)
step 6: r++
step 7: repeat step 2 to step 6 while r < (numRows + frameSize)

Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int toInt(string input)
{
    return stoi(input);
}

class ImageProcessing
{
public:
    int numRows, numCols, minVal, maxVal, newMin, newMax, thrVal;
    int neighborAry[9];
    int CPmasks[8][5][5];
    int neighbor5x5[5][5];
    int **mirror3by3Ary;
    int **mirror5by5Ary;
    int **avgAry;
    int **medianAry;
    int **CPAry;

    void threshold(int **ary1, int **ary2, int frameSize)
    {
        newMin = 0;
        newMax = 1;

        int r = frameSize;
        while (r < numRows + frameSize)
        {
            int c = frameSize;
            while (c < numCols + frameSize)
            {
                if (ary1[r][c] >= thrVal)
                {
                    ary2[r][c] = 1;
                }
                else
                {
                    ary2[r][c] = 0;
                }
                c++;
            }
            r++;
        }
    }
```

```
void imgReformat(int **inAry, ofstream &outImg, int frameSize)
{

    outImg << numRows << " " << numCols << " " << newMin << " " << newMax << endl;
    string str = to_string(newMax);
    int width = str.length();
    int r = frameSize;
    while (r < (numRows + frameSize))
    {
        int c = frameSize;
        while (c < (numCols + frameSize))
        {
            outImg << inAry[r][c];
            str = to_string(inAry[r][c]);
            int ww = str.length();
            while (ww < width)
            {
                outImg << " ";
                ww++;
            }
            c++;
        }
        r++;
        outImg << endl;
    }
}

void loadCPmasks()
{
    int masks[8][5][5] = {{{0, 0, 0, 0, 0},
                           {0, 0, 0, 0, 0},
                           {0, 0, 1, 0, 0},
                           {0, 1, 1, 1, 0},
                           {1, 1, 1, 1, 1}},
                          {{1, 0, 0, 0, 0},
                           {1, 1, 0, 0, 0},
                           {1, 1, 1, 0, 0},
                           {1, 1, 0, 0, 0},
                           {1, 0, 0, 0, 0}},
                          {{1, 1, 1, 1, 1},
                           {0, 1, 1, 1, 0},
                           {0, 0, 1, 0, 0},
                           {0, 0, 0, 0, 0},
                           {0, 0, 0, 0, 0}},
                          {{0, 0, 0, 0, 1},
                           {0, 0, 0, 1, 1},
                           {0, 0, 1, 1, 1},
                           {0, 0, 0, 1, 1},
                           {0, 0, 0, 0, 1}},
                          {{1, 1, 1, 0, 0},
```

```
                             {1, 1, 1, 0, 0},
                             {1, 1, 1, 0, 0},
                             {0, 0, 0, 0, 0},
                             {0, 0, 0, 0, 0}},
                            {{0, 0, 1, 1, 1},
                             {0, 0, 1, 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, 1, 1, 1},
                             {0, 0, 1, 1, 1},
                             {0, 0, 1, 1, 1}},
                            {{0, 0, 0, 0, 0},
                             {0, 0, 0, 0, 0},
                             {1, 1, 1, 0, 0},
                             {1, 1, 1, 0, 0},
                             {1, 1, 1, 0, 0}}};

    memcpy(CPmasks, masks, sizeof(masks));
}

void loadImage(ifstream &input)
{
    mirror3by3Ary = new int *[numRows + 2];
    for (int i = 0; i < numRows + 2; ++i)
    {
        mirror3by3Ary[i] = new int[numCols + 2]();
    }

    mirror5by5Ary = new int *[numRows + 4]();
    for (int i = 0; i < numRows + 4; ++i)
    {
        mirror5by5Ary[i] = new int[numCols + 4];
    }

    for (int i = 0; i < numRows; ++i)
    {
        for (int j = 0; j < numCols; ++j)
        {
            int pixel;
            input >> pixel;
            mirror3by3Ary[i + 1][j + 1] = pixel;
            mirror5by5Ary[i + 2][j + 2] = pixel;
        }
    }
}

void mirrorFraming(int **ary, int frameSize)
```

```
    {
        int totalRows = numRows + 2 * frameSize;
        int totalCols = numCols + 2 * frameSize;

        int yDiff = 1;
        for (int i = frameSize - 1; i >= 0; i--)
        {
            if (frameSize == 1)
            {
                mirror3by3Ary[i] = ary[i + yDiff];
            }
            if (frameSize == 2)
            {
                mirror5by5Ary[i] = ary[i + yDiff];
            }
            yDiff = yDiff + 2;
        }
        yDiff = 1;
        for (int i = totalRows - frameSize; i < totalRows; ++i)
        {

            for (int j = 0; j < totalCols; j++)
            {
                if (frameSize == 1)
                {
                    mirror3by3Ary[i] = ary[i - yDiff];
                }
                if (frameSize == 2)
                {
                    mirror5by5Ary[i] = ary[i - yDiff];
                }
            }
            yDiff = yDiff + 2;
        }

        int xDiff = 1;
        for (int i = frameSize - 1; i >= 0; i--)
        {

            for (int j = 0; j < totalRows; j++)
            {
                if (frameSize == 1)
                {
                    mirror3by3Ary[j][i] = ary[j][i + xDiff];
                }
                if (frameSize == 2)
                {
                    mirror5by5Ary[j][i] = ary[j][i + xDiff];
                }
            }
```

```
            xDiff = xDiff + 2;
        }
        xDiff = 1;
        for (int i = totalCols - frameSize; i < totalCols; ++i)
        {

            for (int j = 0; j < totalRows; j++)
            {
                if (frameSize == 1)
                {
                    mirror3by3Ary[j][i] = ary[j][i - xDiff];
                }
                if (frameSize == 2)
                {
                    mirror5by5Ary[j][i] = ary[j][i - xDiff];
                }
            }
            xDiff = xDiff + 2;
        }
    }

    void computeAvg()
    {
        newMin = 9999;
        newMax = 1;
        avgAry = new int *[numRows + 2];
        for (int i = 0; i < numRows + 2; ++i)
        {
            avgAry[i] = new int[numCols + 2]();
        }

        int r = 1;
        while (r < numRows + 1)
        {
            int c = 1;
            while (c < numCols + 1)
            {
                avgAry[r][c] = avg3x3(r, c);
                if (newMin > avgAry[r][c])
                {
                    newMin = avgAry[r][c];
                }
                if (newMax < avgAry[r][c])
                {
                    newMax = avgAry[r][c];
                }
                c++;
            }
            r++;
        }
```

```
        mirrorFraming(avgAry, 1);
    }

    void computeMedian()
    {
        newMin = 9999;
        newMax = 0;
        medianAry = new int *[numRows + 2];
        for (int i = 0; i < numRows + 2; ++i)
        {
            medianAry[i] = new int[numCols + 2]();
        }

        int r = 1;
        while (r < numRows + 1)
        {
            int c = 1;
            while (c < numCols + 1)
            {
                medianAry[r][c] = median3x3(r, c);
                if (newMin > medianAry[r][c])
                {
                    newMin = medianAry[r][c];
                }
                if (newMax < medianAry[r][c])
                {
                    newMax = medianAry[r][c];
                }
                c++;
            }
            r++;
        }
    }

    void computeCPfilter()
    {
        loadCPmasks();
        newMin = 9999;
        newMax = 0;
        CPAry = new int *[numRows + 4];
        for (int i = 0; i < numRows + 4; ++i)
        {
            CPAry[i] = new int[numCols + 4]();
        }
        int r = 2;
        while (r < numRows + 2)
        {
            int c = 2;
            while (c < numCols + 2)
            {
```

```
            CPAry[r][c] = CP5x5(r, c);
            if (newMin > CPAry[r][c])
            {
                newMin = CPAry[r][c];
            }
            if (newMax < CPAry[r][c])
            {
                newMax = CPAry[r][c];
            }
            c++;
        }
        r++;
    }
}

int CP5x5(int i, int j)
{
    int r = i - 2;

    for (int k = 0; k < 5; k++)
    {
        int c = j - 2;
        for (int l = 0; l < 5; l++)
        {
            neighbor5x5[k][l] = mirror5by5Ary[r][c];
            c++;
        }
        r++;
    }
    int gaussianAvg;
    int leastDiff = 999;
    for (int k = 0; k < 8; k++)
    {
        int convAvg = convolution(CPmasks[k]);
        int diff = abs(mirror5by5Ary[i][j] - convAvg);
        if (diff < leastDiff)
        {
            leastDiff = diff;
            gaussianAvg = convAvg;
        }
    }
    return gaussianAvg;
}

int convolution(int n[][5])
{
    int totalWeight = 0;
    int sumOfProducts = 0;
    for (int i = 0; i < 5; i++)
    {
```

```
            for (int j = 0; j < 5; j++)
            {
                sumOfProducts += n[i][j] * neighbor5x5[i][j];
                totalWeight += n[i][j];
            }
        }
        int result = sumOfProducts / totalWeight;
        return result;
    }

    void sort(int *neighborAry)
    {
        int temp;
        for (int i = 0; i < 9; i++)
        {
            for (int j = i + 1; j < 9; j++)
            {
                if (neighborAry[i] > neighborAry[j])
                {
                    temp = neighborAry[i];
                    neighborAry[i] = neighborAry[j];
                    neighborAry[j] = temp;
                }
            }
        }
    }
    int avg3x3(int i, int j)
    {
        const int frameSize = 1;
        const int totalCols = 2 * frameSize + 1;
        const int totalRows = 2 * frameSize + 1;
        const int totalCells = totalCols * totalRows;
        int sum = 0;
        int r = i - frameSize;
        while (r <= (i + frameSize))
        {
            if (r >= 0 && r < numRows + frameSize)
            {
                int c = j - frameSize;
                while (c <= (j + frameSize))
                {
                    if (c >= 0 && c < numCols + frameSize)
                    {
                        sum += mirror3by3Ary[r][c];
                    }
                    c++;
                }
            }
            r++;
        }
```

```
    int avg = sum / totalCells;
    return avg;
}

int median3x3(int i, int j)
{
    const int frameSize = 1;
    const int totalCols = 2 * frameSize + 1;
    const int totalRows = 2 * frameSize + 1;
    const int totalCells = totalCols * totalRows;

    int r = i - frameSize;

    int index = 0;
    while (r <= (i + frameSize))
    {
        if (r >= 0 && r < numRows + frameSize)
        {
            int c = j - frameSize;
            while (c <= (j + frameSize))
            {
                if (c >= 0 && c < numCols + frameSize)
                {
                    neighborAry[index] = mirror3by3Ary[r][c];
                    index++;
                }
                c++;
            }
        }
        r++;
    }
    sort(neighborAry);
    int median = neighborAry[5];
    return median;
}

void aryToFile(int **ary, ofstream &outFile, int frameSize)
{
    imgReformat(ary, outFile, frameSize);
}

void prettyPrint(int **inAry, ofstream &outFile, int frameSize)
{
    outFile << numRows << " " << numCols << " " << newMin << " " << newMax << endl;
    newMin = 0;
    newMax = 1;

    int r = frameSize;
    while (r < numRows + frameSize)
    {
```

```
            int c = frameSize;
            while (c < numCols + frameSize)
            {
                if (inAry[r][c] > 0)
                {
                    outFile << 1 << " ";
                }
                else
                {
                    outFile << ". ";
                }
                c++;
            }
            outFile << endl;
            r++;
        }
    }

    void cleanUp()
    {
        for (int i = 0; i < numRows + 2; ++i)
        {
            delete[] mirror3by3Ary[i];
        }
        delete[] mirror3by3Ary;

        for (int i = 0; i < numRows + 4; ++i)
        {
            delete[] mirror5by5Ary[i];
        }
        delete[] mirror5by5Ary;

        for (int i = 0; i < numRows + 2; ++i)
        {
            delete[] avgAry[i];
        }
        delete[] avgAry;

        for (int i = 0; i < numRows + 2; ++i)
        {
            delete[] medianAry[i];
        }
        delete[] medianAry;

        for (int i = 0; i < numRows + 4; ++i)
        {
            delete[] CPAry[i];
        }
        delete[] CPAry;
    }
```

```
};

int main(int argc, const char *argv[])
{
    //READ
    string inputName = argv[1];
    ifstream input;
    input.open(inputName);

    int thrVal = toInt(argv[2]);

    //WRITES
    string rfImgName = argv[3], avgOutImgName = argv[4], avgThrImgName = argv[5],
avgPrettyPrintName = argv[6], medianOutImgName = argv[7], medianThrImgName = argv[8],
medianPrettyPrintName = argv[9], CPOutImgName = argv[10], CPThrImgName = argv[11],
CPPrettyPrintName = argv[12];
    ofstream rfImg, AvgOutImg, AvgThrImg, AvgPrettyPrint, MedianOutImg, MedianThrImg,
MedianPrettyPrint, CPOutImg, CPThrImg, CPPrettyPrint;
    rfImg.open(rfImgName);
    AvgOutImg.open(avgOutImgName);
    AvgThrImg.open(avgThrImgName);
    AvgPrettyPrint.open(avgPrettyPrintName);
    MedianOutImg.open(medianOutImgName);
    MedianThrImg.open(medianThrImgName);
    MedianPrettyPrint.open(medianPrettyPrintName);
    CPOutImg.open(CPOutImgName);
    CPThrImg.open(CPThrImgName);
    CPPrettyPrint.open(CPPrettyPrintName);

    if (input.is_open())
    {

        if (rfImg.is_open() && AvgOutImg.is_open() && AvgThrImg.is_open() &&
AvgPrettyPrint.is_open() && MedianOutImg.is_open() && MedianThrImg.is_open() &&
MedianPrettyPrint.is_open() && CPOutImg.is_open() && CPThrImg.is_open() &&
CPPrettyPrint.is_open())
        {
            ImageProcessing imgProcessing;

            imgProcessing.thrVal = thrVal;

            input >> imgProcessing.numRows >> imgProcessing.numCols >> imgProcessing.minVal
>> imgProcessing.maxVal;

            imgProcessing.newMin = imgProcessing.minVal;

            imgProcessing.newMax = imgProcessing.maxVal;

            imgProcessing.loadImage(input);
```

```
            imgProcessing.mirrorFraming(imgProcessing.mirror3by3Ary, 1);

            imgProcessing.imgReformat(imgProcessing.mirror3by3Ary, rfImg, 1);

            imgProcessing.computeAvg();

            imgProcessing.imgReformat(imgProcessing.avgAry, AvgOutImg, 1);

            int **thrAry;
            thrAry = new int *[imgProcessing.numRows + 2];
            for (int i = 0; i < imgProcessing.numRows + 2; ++i)
            {
                thrAry[i] = new int[imgProcessing.numCols + 2]();
            }

            imgProcessing.threshold(imgProcessing.avgAry, thrAry, 1);

            imgProcessing.aryToFile(thrAry, AvgThrImg, 1);

            imgProcessing.prettyPrint(thrAry, AvgPrettyPrint, 1);

            imgProcessing.computeMedian();

            imgProcessing.imgReformat(imgProcessing.medianAry, MedianOutImg, 1);

            imgProcessing.threshold(imgProcessing.medianAry, thrAry, 1);

            imgProcessing.aryToFile(thrAry, MedianThrImg, 1);

            imgProcessing.prettyPrint(thrAry, MedianPrettyPrint, 1);

            imgProcessing.mirrorFraming(imgProcessing.mirror5by5Ary, 2);

            imgProcessing.computeCPfilter();

            imgProcessing.imgReformat(imgProcessing.CPAry, CPOutImg, 2);

            imgProcessing.threshold(imgProcessing.CPAry, thrAry, 1);

            imgProcessing.aryToFile(thrAry, CPThrImg, 2);

            imgProcessing.prettyPrint(thrAry, CPPrettyPrint, 2);

            // imgProcessing.cleanUp();
        }
        else
        {
            cout << "Error: Some output files is missing or couldnt be opened" << endl;
        }
    }
```

```
    else
    {
        cout << "Error: Reading the input file: " << inputName << endl;
    };

    rfImg.close();
    AvgOutImg.close();
    AvgThrImg.close();
    AvgPrettyPrint.close();
    MedianOutImg.close();
    MedianThrImg.close();
    MedianPrettyPrint.close();
    CPOutImg.close();
    CPThrImg.close();
    CPPrettyPrint.close();
    return 0;
}
```

Outputs:

rfImg File:

```
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 4 551 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 434 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
3 1 2 3 445 1 423 4 45512 3 4 5 1 2 3 4 5 1 2 584 5 1 2 534 5 1 2 3 4 45112 434 5 412 3 4 5
4 1 2 3 4 5 1 2 3 4 55512 3 4 5 1 2 3 4 5 1 2 584 5 1 2 634 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
5 1 623 4 5 1 2 434 5 1 2 3 4 5 1 2 3 4 5 1 2 584 5 1 2 534 351 2 3 4 5 412 3 4 5 1 2 3 4 5
6 1 2 3 4 5 1 2 3 445 1 2 3 4 5 1 2 3 4 5 1 2 414 5 1 2 3 445 1 2 3 4 5 512 3 4 55512 3 4 5
7 1 2 3 4 5 1 2 3 445 1 2 3 4 5 8 2 3 4 5 1 2 8 4 5 1 123 445 1 2 3 4 5 1 2 614 5 1 2 3 4 5
8 1 2 3 4 5 1 2 5 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 445 1 2 3 4 451 2 3 4 551 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 4838485 1 2 3 445 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48244848481 2 434 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 4 5 112 3 4 5 1 2 3 4 5 1 2 3 4 48334 344148482 3 445 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
2 1 2 3 4 45512 3 4 5 1 2 3 4 5 1 2 3 4848484 8 484848483 4 5 192 3 4 5 1 2 3 145 1 2 3 4 5
3 1 2 3 4 55512 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 3 445 1 2 3 4 5
4 413233343738393130323435343538404860360484138353432313028252824222018 8 6 134 5 1 2 3 145
5 1 213 4 5 1 2 3 4 5 1 2 3 4 5 1 484848481048484834484848485 1 2 3 4 5 1 2 3 4 5 1 323 4 5
6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 484848484848484848484848484 48481 2 3 4 5 1 2 3 4 551 2 3 4 5
7 1 2 3 145 1 2 3 4 5 1 2 3 4 484848142434142434 484846484848482 3 4 511 2 3 4 5 1 2 3 4 5
8 1 2 3 4 5 1 2 3 4 5 1 2 3 48414844488 45484 484848484848484 4 483 4 5 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 151 123 4 5 1 2 484848486048484848486162484848488 7 4848484 5 1 2 3 4 5 1 2 134 5
101 2 3 4 5 1 2 3 4 5 1 4848485 4848483 48484848486 484847488 484848485 1 123 4 5 1 2 3 4 5
1 1 523 4 5 1 123 4 5 484858484848484848282384848484848488 4848282838281 81 2 3 4 5 112 3 4 5
2 1 2 3 4 5 1 2 3 4 484858484848404847484848414842485248 4 8 5 4848483838281 83 4 5 1 2 3 145
3 61222324273829313032343534353840486036048413835343231302835284432308 8 16434 5 1 2 3 4 5
4 1 2 3 4 5 1 2 4848484848484 484848485858583838584858582824144848484838384338184 5 1 2 3 4 5
5 1 2 4841484248438 486048484848414248434846484548404843 3 48304848488 484838384 2 8 8 8 4 5
6 1 2 3 4 5 1 2 4848484848488 48486348634 4848484 4848488 4 484848484818484848484 5 1 2 3 4 5
7 1 2 3 4 5 1 2 3 48488 484842484818484848486348484848484888 8 4848485 48488 3 4 5 1 2 3 4 5
8 1 2 3 4 5 132 3 4 484862485548484484 7 8 484848544858484484 4 8 484848484823 4 5 112 3 4 5
9 1 2 3 4 5 1 2 3 4 5 484848484848488288 48484 4848481 48486 4 8 4 4848481 2 3 4 5 1 2 3 4 5
101 2 3 4 5 1 2 3 4 5 1 484848483 484848484848184848484848484888 4 48485 1 2 3 4 5 1 123 4 5
1 21222324272829313032343534353840486036048413835343231302825284222201 88 6 3 4 5 1 2 3 145
2 1 2 3 4 5 1 2 3 4 5 1 2 484848484818484848484848484848488 48488 4 484 5 1 2 3 14151 2 3 4 5
3 112 3 4 5 1 2 3 4 5 1 2 3 484848484848484848484848484848488 4 3 4 5 1 2 3 4 5 1 2 3 4 5
4 1 2 3 4 151 2 3 4 5 1 2 3 4 484841424348404842484348444824848482 3 4 5 1 2 3 4 551 2 3 4 5
5 1 2 3 42551 423 4 5 1 2 3 4 5 3444413424343441343442343424 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 4848584 1 28411 482 4 8 485 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 5 132 3 4 5 1 2 48488 4834354148488 484 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
8 1 2 3 4 511 2 3 4 5 1 2 3 4 5 1 2 3 483848388 1 4838483 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
9 1 2 3 4 5 1 123 4 5 1 2 3 4 5 1 2 3 4 484848484848482 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
101 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48481848481 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 1 2 3 445 1 2 3 4 5 1 123 4 5 1 2 3 4 5 1 4848485 1 2 3 4 5 1 2 3 4 551 2 3 4 551 2 3 4 5
2 1 2 3 485 1 2 3 4 5551123 4 5 1 2 3 4 5 1 42484 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 45512 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 484 5 1 2 3 4 5 1 2 3 635 1 2 3 4 5 1 2 3 4 5
4 1 2 3 4 5 1 2 3 4 5 1 2 3 145 1 2 3 4 5 1 2 484 5 1 2 3 4 5 1 2 595 5 1 2 434 5 1 2 334 5
5 1 2 3 4 5 112 3 445 1 2 3 4 5 1 2 3 4 5 1 2 484 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
```

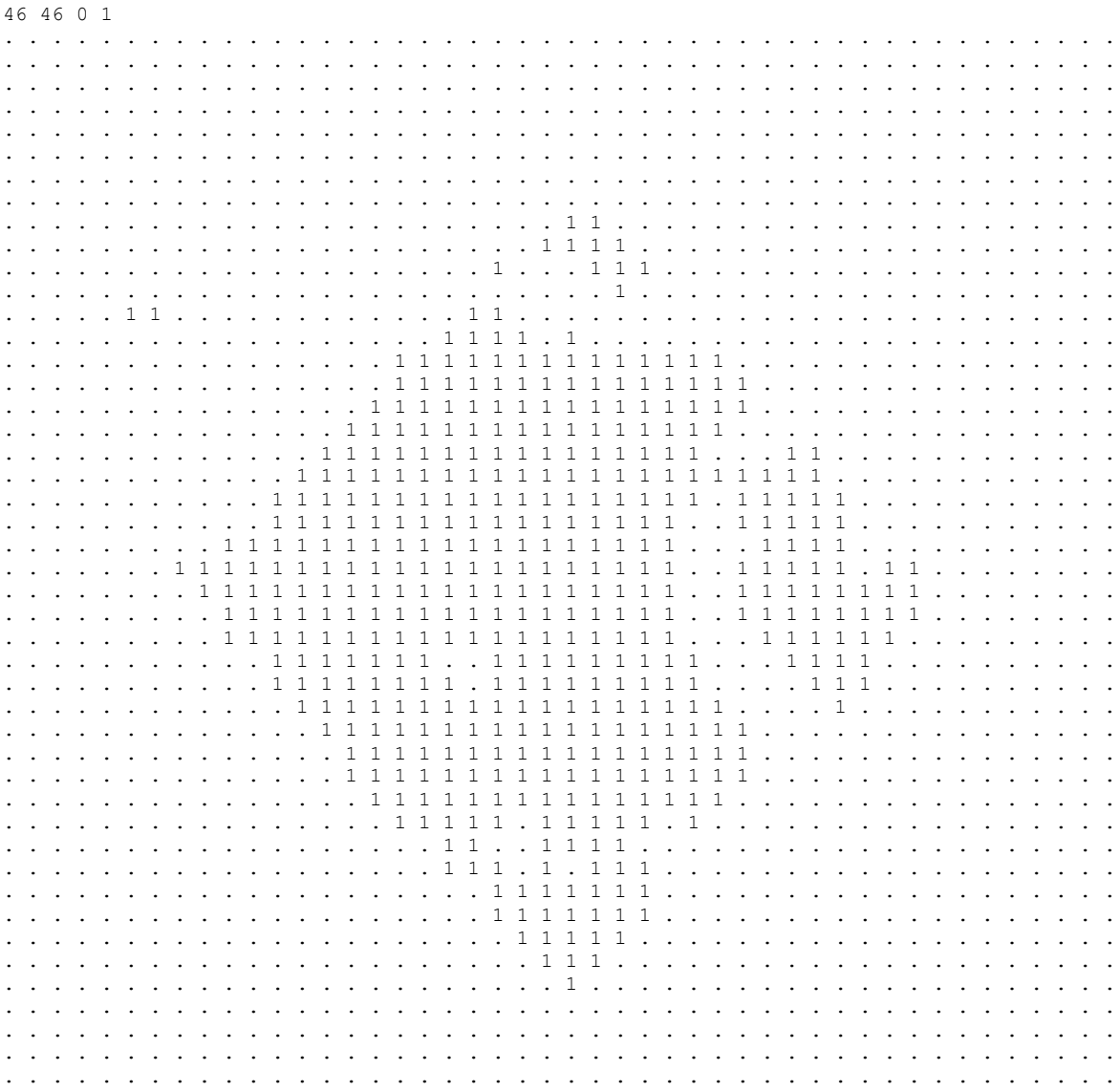
AvgOutImg File:

```
46 46 1 54
1 1 2 3 9 8 8 2 3 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3 2 6 7 8 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3
1 1 2 7 1413126 7 8 13127 3 4 3 2 2 3 4 3 2 8 9 103 2 1213143 2 2 3 8 8 8 7 7 8 7 7 6 3 4 3
2 2 2 7 1413126 7 142423133 4 3 2 2 3 4 3 2 1415163 2 1819203 2 2 3 8 8 8 7 7 8 7 7 6 3 4 3
3 9 8 148 7 7 1011182423133 4 3 2 2 3 4 3 2 2021223 2 1920256 6 2 3 8 1312127 8 7 7 6 3 4 3
3 9 8 9 4 3 2 6 111818137 3 4 3 2 2 3 4 3 2 1819203 2 141924116 2 3 4 1312123 9 14137 3 4 3
4 9 8 9 4 3 2 6 1617122 2 3 4 4 3 2 3 4 3 2 1213143 3 8 1821156 2 3 4 1312189 1614137 3 4 3
5 3 2 3 4 3 2 2 1213122 2 3 4 4 3 2 3 4 3 2 6 12137 3 3 1317167 2 3 9 1413149 1614137 3 4 3
5 3 2 3 4 3 2 2 7 8 7 2 2 3 4 4 3 2 3 4 3 7 112117123 3 1317167 2 3 9 8 8 8 9 103 2 2 3 4 3
4 3 2 3 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 8 15233331227 6 1117127 2 3 9 8 8 2 3 4 3 2 2 3 4 3
2 2 2 3 4 4 3 3 3 4 3 2 2 3 4 3 2 2 3 8 162430373932171116171 22 2 3 4 3 2 2 3 4 3 2 2 3 4 3
1 1 2 3 8 14138 3 4 3 2 2 3 4 3 2 2 7 183129272841423222171 29 4 4 3 3 4 3 2 2 4 5 4 2 2 3 4 3
1 1 2 3 142524143 4 3 2 2 3 4 3 2 2 8 18272216172633291913 99 4 4 3 3 4 3 2 2 8 9 8 2 2 3 4 3
6 9 131324353524131312131213141415172633372923212429292418121312119 9 7 5 4 10108 2 2 4 5 4
7 121515192525181313121312131414202836383330283027272729282216109 9 9 7 5 4 9 9 7 6 5 7 5 4
8 121515141414131313121312131420314351524644414442414137373125159 9 9 7 5 4 4 108 115 7 5 4
4 5 4 6 5 4 2 2 3 4 3 2 2 3 8 183342464641414141414146424238321 77 3 9 8 7 2 3 9 8 115 6 4 3
5 3 2 4 5 4 2 2 3 4 3 2 2 7 173242464141414041374243474242383322128 9 8 7 2 3 9 8 8 2 3 4 3
5 3 2 4 6 5 4 3 4 4 3 2 7 173242484843414140424045444747433429282818148 7 2 3 4 3 2 3 4 5 3
6 4 2 3 5 4 4 3 4 4 3 7 173237424348393838424446464443474329242938331 88 3 3 4 4 3 2 3 4 5 3
4 8 7 8 5 4 6 4 5 4 8 183344444344494440394448514644434339303234423726125 3 4 4 4 3 4 4 5 3
3 8 7 8 4 3 3 3 4 8 183445504442424742403943466424343393424323442413422137 5 4 4 3 3 4 5 4
8 161815101213131317283945454543454750504744444343434333282330343837302115131 08 4 3 3 4 5 4
9 10139 101212172132374444393938454752545448444243454537281926354341352926231 6103 2 2 4 5 4
9 111818242526313137394444383938444750545349444342443932282633384338332931322213 3 3 4 4 4 3
3 2 7 121716172632434449493838374848514647444641414444332523344248423837404026143 3 4 4 4 3
4 3 7 121716172127384044444242424444464444444944414242332422324148383535393621123 3 4 4 4 3
5 3 2 3 4 4 4 8 173338454545434346413830354049454344494429202034434339393528139 4 3 3 3 4 3
5 3 2 3 4 4 4 3 7 182940454948484437282734404445484443433419102034384338281 33 4 4 3 3 3 4 3
6 4 2 3 4 4 4 3 3 8 183444504843433531273438394045444343382815152438433217 7 3 4 4 4 4 4 4 3
7 7 8 9 101111111213182838434339414144454643373640383737373122172129332110 3 3 4 3 3 3 5 5 4
5 6 8 9 101111111213121827384339414047495148424140434237373632212124241 25 3 4 6 5 4 3 5 5 4
5 7 9 9 101111111213121317283844464547495148464443434237373632211614147 5 3 4 6 5 3 2 4 5 4
3 3 3 3 5 4 3 2 3 4 3 2 7 173343474342434647464746474743404136241 48 9 3 2 2 4 11119 2 3 4 3
4 3 3 7 1414136 7 4 3 2 2 7 183140434341404142434243434339362818 8 3 4 3 2 2 3 9 8 8 2 3 4 3
3 2 2 7 1414136 7 4 3 2 2 3 8 173038443831293435363333293027231 27 3 4 3 2 2 3 9 8 8 2 3 4 3
4 3 2 7 1313126 7 4 4 4 3 3 4 7 162941342823323235339252320102 2 3 4 3 2 2 3 4 3 2 2 3 4 3
5 3 2 3 9 8 7 2 3 4 4 4 3 3 4 3 7 173333332731253030322319148 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3
5 3 2 3 9 8 8 3 4 4 4 4 3 3 4 3 2 7 172737393933364037271 39 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3
6 4 2 3 9 8 8 3 4 4 3 2 2 3 4 3 2 2 7 17324139333536311 68 4 3 2 2 3 4 3 2 2 3 4 3 2 2 3 4 3
4 3 2 7 8 7 3 3 4 4 3 3 3 4 4 3 2 2 3 8 183339443932177 3 4 3 2 2 3 9 8 8 2 3 9 8 8 2 3 4 3
3 2 2 1213122 2 3 9 14169 5 4 3 2 2 3 4 8 22333930177 2 3 4 3 2 2 3 9 8 8 2 3 9 8 8 2 3 4 3
1 1 2 121722127 3 9 14169 5 4 3 2 2 3 4 3 122632238 2 2 3 4 3 2 2 9 16158 2 3 9 8 8 2 3 4 3
2 2 2 7 1318127 3 9 14148 5 5 4 2 2 3 4 3 7 2122193 2 2 3 4 3 2 8 1516102 6 7 8 3 2 5 6 7 3
3 2 2 3 8 14138 7 8 7 2 2 4 5 4 2 2 3 4 3 2 1718193 2 2 3 4 3 2 8 1516102 6 7 8 3 2 5 6 7 3
3 2 2 3 4 4 3 3 7 8 7 2 2 4 5 4 2 2 3 4 3 2 1213143 2 2 3 4 3 2 8 9 103 2 6 7 8 3 2 5 6 7 3
2 1 1 2 2 3 2 2 6 7 6 1 1 2 2 2 1 1 2 2 2 1 6 7 7 2 1 1 2 2 2 1 1 2 2 2 1 1 2 2 2 1 1 2 2 2
```

AvgThrImg file:

```
46 46 0 1
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000011000000000000000000000000000
0000000000000000000000111100000000000000000000000000
0000000000000000000010001110000000000000000000000000
0000000000000000000000001000000000000000000000000000
0000011000000000000011000000000000000000000000000000
0000000000000000000111101000000000000000000000000000
0000000000000000111111111111110000000000000000000000
0000000000000000111111111111110000000000000000000000
0000000000000000111111111111110000000000000000000000
0000000000000000111111111111110000000000000000000000
0000000000000011111111111111110001100000000000000000
0000000000000111111111111111111111000000000000000000
0000000000001111111111111111101111100000000000000000
0000000000001111111111111111100111110000000000000000
0000000000111111111111111111100011110000000000000000
0000000011111111111111111111100111111011000000000000
0000000011111111111111111111100111111100000000000000
0000000011111111111111111111100111111100000000000000
0000000011111111111111111111100011111000000000000000
0000000000111111100111111111000111100000000000000000
0000000000111111101111111111000011100000000000000000
0000000000111111111111111111000010000000000000000000
0000000000001111111111111111100000000000000000000000
0000000000001111111111111111100000000000000000000000
0000000000001111111111111111100000000000000000000000
0000000000001111101111101000000000000000000000000000
0000000000000000110011110000000000000000000000000000
0000000000000000111010111000000000000000000000000000
0000000000000000001111111000000000000000000000000000
0000000000000000001111111000000000000000000000000000
0000000000000000000111110000000000000000000000000000
0000000000000000000011100000000000000000000000000000
0000000000000000000010000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
```

AvgPrettyPrint file:



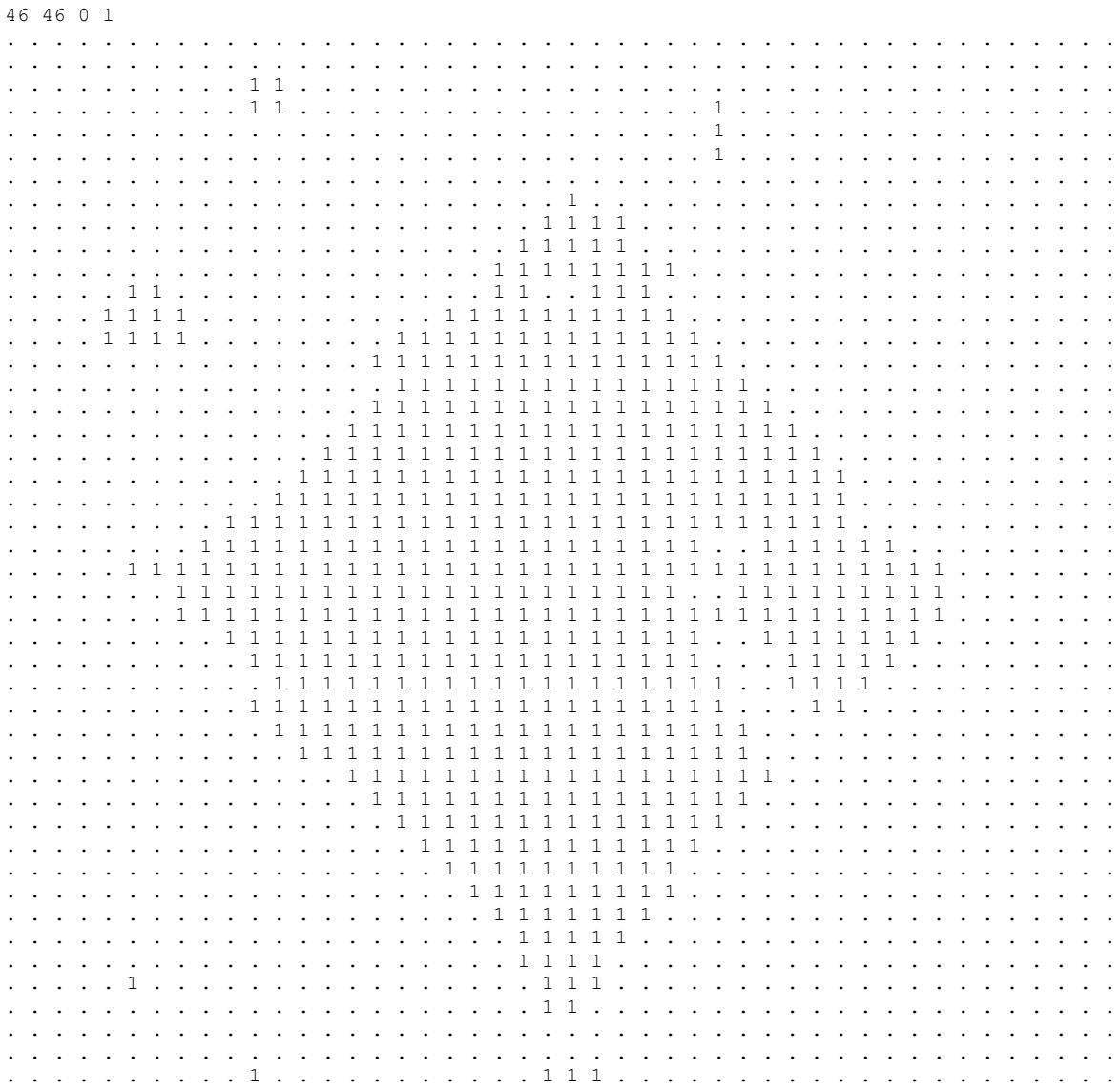
MedianOutImg File:

```
46 46 1 58
1 2 2 3 5 8 5 3 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 3 3 6 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
1 2 2 3 5 5 5 3 4 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 5 5 3 4 5 5 5 3 3 4 5
2 2 2 3 5 5 5 3 4 4 45453 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 5 4 2 2 3 4 5 5 3 4 5 5 5 3 3 4 5
3 3 3 4 5 5 5 3 4 5 45453 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 354 2 2 3 4 5 5 3 4 5 5 5 3 3 4 5
4 4 3 4 4 4 2 2 4 5 5 5 3 3 4 4 2 2 3 4 4 2 2 4 5 4 2 2 4 355 2 2 3 4 5 5 3 3 4 5 5 3 3 4 5
5 5 3 4 4 4 2 2 4 5 5 2 2 3 4 5 5 3 3 4 4 2 2 4 5 4 5 3 12355 2 2 3 4 5 5 3 4 5 5 5 3 3 4 5
5 2 2 3 4 4 2 2 4 5 5 2 2 3 4 5 5 3 3 4 4 2 2 4 5 5 5 3 4 5 5 2 2 3 4 5 5 3 4 5 5 5 3 3 4 5
5 2 2 3 4 4 2 2 4 5 5 2 2 3 4 5 5 3 3 4 4 5 3 388 5 5 3 4 5 5 2 2 3 4 4 2 2 4 5 4 2 2 3 4 5
4 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 3 4 5 5 384848442 2 4 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
1 2 2 3 4 5 5 3 3 4 4 2 2 3 4 4 2 2 3 4 5 33384848485 3 4 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
1 2 2 3 4 5 5 3 3 4 4 2 2 3 4 4 2 2 3 5 484834414848484434 5 5 5 3 3 4 4 2 2 3 5 5 2 2 3 4 5
1 2 2 3 4 45453 3 4 4 2 2 3 4 4 2 2 4 8 48338 8 4148488 8 5 5 5 3 3 4 4 2 2 3 5 5 2 2 3 4 5
4 3 3 4 344545384 5 5 5 3 4 5 5 5 4 484848483835353534312 88 1919194 5 5 5 3 6 5 5 2 2 3 5 5
6 5 2121333738314 5 5 5 3 4 5 5 384848484848414135343432312 8255 3 4 5 5 5 3 4 5 5 5 3 4 5 5
6 6 21215 5 5 3 4 5 5 5 3 4 5 354848484848484848484848484848484828243 4 5 5 5 3 4 5 4 5 3 4 5 5
5 5 3 4 5 5 2 2 3 4 4 2 2 3 4 5 484848484848484848484848484848485 3 3 4 4 2 2 3 4 4 5 3 4 4 5
5 2 2 3 5 5 2 2 3 4 4 2 2 3 5 484848484848484848484848484848483 4 4 4 2 2 3 4 4 2 2 3 4 5
6 2 2 3 5 5 5 3 4 4 4 2 2 4 484848484848484848484848484848484484 5 4 2 2 3 4 4 2 2 4 5 5
6 2 2 3 4 4 5 3 4 4 4 5 3 484848484848484848484848484848484848485 5 5 3 4 4 4 2 2 4 5 5
4 2 3 4 4 4 5 3 4 4 5 5 484848484848484848484848484848484848484848385 5 3 4 4 5 5 3 4 5 5
3 2 3 4 4 4 5 3 4 4 5 484848484848484848484848484848484848474848484838281 83 4 4 5 5 3 3 5 5
3 3 22225 5 12121230344848484848484848484848484848484848313035444438382818164 5 5 5 3 3 5 5
8 3 3 4 5 5 5 2930484848484848484848485858484841484848483028284848483838383828185 4 2 2 3 5 5
5 4 22232738384243484848484848484848485858484645454848323030354848483838383838385 4 5 3 4 5 5
4 2 2 4 5 5 5 434848484848484848484848585848484848484848282848484848484848385 4 5 3 4 5 5
5 2 2 4 5 5 5 424848484848484848484848484848484848484848483048484848484848385 4 5 3 4 5 5
5 2 2 3 4 5 5 3 4 484848484848484848484848484848484848484848488 8 4848484848484484 5 5 5 3 3 4 5
6 2 2 3 4 5 5 3 3 5 484848484848484848484848484848484848488 8 8 48484848483 4 4 5 5 3 3 4 5
7 2 2 3 4 5 5 3 3 4 5 484848484848484848484848484848484848488 8 484848485 3 3 4 5 5 3 4 4 5
7 9 3 4 5 5 5 3 4 5 3035484848484848484848484848484848484848482824244848206 3 4 4 4 5 3 4 5 5
5 2 3 4 5 5 5 3 4 5 5 32354848484848484848484848484848484848484848252424228 5 3 4 5 5 5 3 4 5 5
5 3 114 5 5 5 3 4 5 5 5 34354848484848484848484848484848484848484848252220185 5 3 4 5 5 2 2 3 5 5
4 2 3 3 4 4 2 2 3 4 4 2 2 4 484848484848484848484848484848484848488 4 5 4 2 2 3 5 5 2 2 3 4 5
4 3 3 3 5 5 5 3 4 4 4 2 2 3 5 484848444848484848484848484848484848488 3 4 4 4 2 2 3 4 4 2 2 3 4 5
4 2 2 3 5 5 5 3 4 4 4 2 2 3 4 5 444444434034414142434343423434284 3 3 4 4 2 2 3 4 4 2 2 3 4 5
5 2 2 3 5 5 5 3 4 4 5 5 3 3 4 5 5 444848343435354142423434245 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
5 2 2 3 4 4 2 2 3 4 5 5 3 3 4 4 5 3 484848383835414848388 5 5 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
6 2 2 3 4 4 5 3 4 4 5 5 3 3 4 4 2 2 4 484848484841484848484 5 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
6 2 2 3 4 4 5 3 4 4 4 2 2 3 4 4 2 2 3 5 484848484848483 4 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
4 2 2 3 5 5 5 3 4 4 4 5 3 4 4 4 2 2 3 4 5 48484848485 3 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
2 2 2 3 5 5 2 2 3 4 5 123 4 4 4 2 2 3 4 5 4248848485 2 2 3 4 4 2 2 3 4 4 2 2 3 4 4 2 2 3 4 5
2 2 2 3 5 445 3 3 4 5 123 4 4 4 2 2 3 4 4 5 4848485 2 2 3 4 4 2 2 3 5 5 2 2 3 4 4 2 2 3 4 5
2 2 2 3 5 5 5 3 3 4 5 5 3 4 5 5 2 2 3 4 4 5 42425 4 2 2 3 4 4 2 2 4 5 5 2 2 4 5 4 2 2 4 5 5
3 2 2 3 4 5 5 3 3 5 5 2 2 3 5 5 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 4 5 5 2 2 4 5 4 2 2 4 5 5
3 2 2 3 4 5 5 3 3 5 5 2 2 3 5 5 2 2 3 4 4 2 2 4 5 4 2 2 3 4 4 2 2 4 5 5 2 2 4 5 4 2 2 4 5 5
5 5 3 4 5 6 6 111111335 3 4 5 5 5 3 4 5 5 5 4444445 5 3 4 5 5 5 3 4 5 5 5 3 4 5 5 5 3 4 5 5
```

MedianThrImg File:

```
46 46 0 1
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000110000000000000000000000000000000000000000
0000000000110000000000000000010000000000000000000000
0000000000000000000000000000010000000000000000000000
0000000000000000000000000000010000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000010000000000000000000000000000
0000000000000000000001111000000000000000000000000000
0000000000000000000011111000000000000000000000000000
0000000000000000000111111110000000000000000000000000
0000011000000000000011001110000000000000000000000000
0000111100000000000111111111000000000000000000000000
0000111100000000011111111111100000000000000000000000
0000000000000001111111111111110000000000000000000000
0000000000000001111111111111110000000000000000000000
0000000000000011111111111111110000000000000000000000
0000000000000011111111111111110000000000000000000000
0000000000001111111111111111111000000000000000000000
0000000000001111111111111111111000000000000000000000
0000000001111111111111111111111100000000000000000000
0000000011111111111111111111100111111000000000000000
0000011111111111111111111111111111111110000000000000
0000001111111111111111111111100111111111000000000000
0000001111111111111111111111111111111110000000000000
0000000111111111111111111111100111111100000000000000
0000000111111111111111111111100111111000000000000000
0000000111111111111111111111100011110000000000000000
0000000111111111111111111111100111100000000000000000
0000000111111111111111111111100011000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000000000011111111111111111110000000000000000000000
0000100000000000000000011100000000000000000000000000
0000000000000000000000011000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000100000000000111000000000000000000000000000
```

MedianPrettyPrint File:



CPOutImg File:

```
46 46 1 55
1 1 2 2 3 6 2 3 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 3 4 3 2 4 3 7 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
2 1 2 3 3 142 2 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 203 3 2 2 3 3 4 2 2 3 3 6 2 2 3 4 4
3 1 2 3 146 6 133 3 23242 3 3 4 2 2 3 3 4 2 2 223 3 2 2 253 6 2 2 3 3 12122 123 6 8 2 3 4 4
3 1 2 2 3 6 2 2 6 7 23242 3 3 4 2 2 3 3 4 2 2 203 7 2 2 243 6 2 2 3 3 4 6 3 7 3 6 2 3 3 4 4
4 2 143 3 6 2 2 23117 2 2 3 4 4 2 2 3 3 4 3 2 223 5 3 2 256 292 2 3 3 4 193 8 2 7 2 3 3 4 4
5 2 2 3 3 4 2 2 2 238 2 2 3 4 4 2 2 3 3 4 2 2 203 7 3 2 3 236 2 2 3 3 4 143 3 3 16152 3 4 4
8 2 3 3 3 4 2 2 3 163 2 2 3 4 4 4 2 3 3 4 3 2 9 3 3 3 133 216 2 2 3 3 4 8 2 183 3 2 2 3 4 4
5 2 2 3 3 4 2 2 4 3 4 2 2 3 4 4 2 2 3 3 4 3 2 6 337 3 3 3 3 212 2 3 3 142 2 3 3 4 2 2 3 4 4
5 1 2 3 4 4 2 2 3 3 4 2 2 3 4 4 2 2 3 3 4 3 3739403 3 3 3 177 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
1 1 2 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 3 4 32244142426 2 224 4 2 3 3 3 4 2 2 3 4 5 2 2 3 4 4
1 1 2 3 4 4 9 2 3 3 4 2 2 3 3 4 2 2 3 3 293010323739421l8 175 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
1 1 2 3 5 37373 3 3 5 2 2 3 3 4 2 2 3 383737212342414237117 6 172 3 5 5 2 2 3 105 2 2 3 4 5
1 1 2 3 3 26263 4 3 4 2 2 3 3 4 2 2 2 8 18161716172419139 7 4 4 3 3 4 5 2 2 4 9 5 2 2 3 4 5
3 151524353535352413131314152031435152515246434132333131313022251511 9 9 8 6 154 5 2 2 3 7 5
5 3 154 4 5 2 2 3 3 4 2 2 2 3 8 1146464646214141413842464242 7 3 7 4 5 2 3 3 4 5 5 123 4 5
7 2 2 4 4 5 2 2 3 3 5 2 2 2 6 7 485147464746444344474747304241 4 5 8 7 5 2 2 3 4 112 2 3 4 5
5 2 3 4 6 5 2 2 3 4 4 2 2 3 3 484848414343414342374749464743433 82 3 3 142 2 3 3 5 3 2 3 4 5
6 2 2 3 4 5 2 2 3 4 4 2 2 3 474248434838454637464847474747472212433 7 3 2 2 3 4 4 2 2 3 4 5
9 2 3 4 4 6 3 6 3 4 4 2 2 4944484849484844484648514747474822284240422 5 2 3 3 4 4 2 2 5 4 5
8 2 2 3 4 5 3 3 3 4 4 2 4950493748474837434950464742484847432442414244 4 2 123 4 4 2 2 3 4 5
4 4 214 4 5 4 134 4 6 484745454849484949403947484647444323394330283830153 3 3 4 4 5 2 3 4 5
3 3 7 4 3 4 3 3 3 4 464750454746404947494849424642454545192619434143353829184 4 5 2 2 3 6 5
2 181824242631293131373938393839394754535448434241393228302834264132291513153 24 4 3 3 3 4 5
3 3 2 7 9 101212434647494949374848484854545538414645454628251948484839384336153 3 2 2 3 4 5
4 3 2 24252631414125474747444644424448444847494446404817224031484148284036373 93 3 4 4 5 4 4
5 2 2 3 4 4 4 3 434449484949374848514852294848474149494719204848434848324043414 5 3 3 3 4 4
5 2 3 3 4 4 3 3 3 444529494842484828494644494948454844484310104841483343389 3 4 4 3 3 3 4 5
6 2 2 3 4 4 9 2 3 3 454950485048434824312749454845494449441515104843434343 2 3 4 4 5 2 3 4 4
102 2 3 4 5 3 4 3 5 6 484948484848442726514936485048344443101710134343452 3 3 4 4 3 3 3 4 5
7 2 2 3 4 4 3 3 3 3 8 9 504850483347474948484839494545434344361515434 85 2 3 3 4 5 3 5 3 4 5
4 9 10111111111213182833363836383947525149484138363737323131222526211914 7 7 3 4 5 2 2 3 6 5
3 3 3 3 4 5 2 3 3 3 4 2 2 454347454738484850484647474747364541148 254 5 2 2 3 11112 3 3 4 5
3 9 3 3 4 5 5 6 4 3 5 2 2 3 454645474849484748464545444445393 78 3 3 3 5 2 2 3 4 5 2 2 3 4 5
3 2 2 3 4 142 2 3 3 4 2 2 3 3 484442434447434843484347444330413 82 3 3 4 2 2 3 4 112 2 3 4 4
4 2 2 3 14142 143 4 4 2 2 3 3 4 3145413323323642333543383323 2 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
5 2 2 2 3 6 2 3 3 4 4 2 2 3 3 4 2 464344272929412542211914363 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
5 2 3 3 3 3 2 3 3 4 4 5 2 3 3 4 2 2 42412340333540374013313 3 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
6 2 2 3 3 9 2 2 3 4 4 2 2 3 3 4 2 2 3 3741393826253536463 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
5 2 2 3 3 7 3 9 3 4 4 2 3 3 4 4 2 2 3 3 453944414439422 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
6 1 2 2 3 3 2 2 3 4 4 2 2 3 4 4 2 2 3 3 4 41412942402 2 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
1 1 2 3 224 2 3 3 4 4 2 103 4 4 2 2 3 3 4 3 4444448 2 2 3 3 4 2 2 3 3 162 2 3 3 9 2 2 3 4 4
2 2 2 3 187 2 2 3 3 1615144 4 5 2 2 3 3 4 3 39443 7 2 2 3 3 4 2 2 3 3 8 2 2 3 3 7 2 2 3 4 6
3 1 2 3 4 17232 3 3 7 2 2 4 4 5 2 2 3 3 4 3 2 323 8 2 2 3 3 4 2 2 2 163 2 2 2 3 3 2 2 3 4 4
3 1 2 3 4 4 2 3 3 3 7 2 2 3 5 5 2 2 3 3 6 2 2 223 3 2 2 3 3 4 2 2 173 3 2 2 8 3 3 2 2 7 4 4
4 2 2 3 4 4 132 3 7 3 2 2 3 4 5 2 2 3 3 4 2 2 193 3 2 2 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
4 2 2 3 4 4 2 3 3 2 7 2 2 3 4 5 2 2 3 3 4 3 2 4 3 8 2 2 3 3 4 2 2 3 3 4 2 2 3 3 4 2 2 3 4 4
```

CPThrImg File:

```
46 46 0 1
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000010000000000000000000000000000
0000000000000000000000111000000000000000000000000000
0000000000000000000001011100000000000000000000000000
0000000000000000000001011110000000000000000000000000
0000011000000000000011100111100000000000000000000000
0000000000000000000000000000000000000000000000000000
0000111100000001111111111111110000000000000000000000
0000000000000000011110111111110000000000000000000000
0000000000000001111111111111110000000000000000000000
0000000000000011111111111111110000000000000000000000
0000000000000011111111111111110010000000000000000000
0000000000000111111111111111100111000000000000000000
0000000000001111111111111111101111000000000000000000
0000000000011111111111111111011101100000000000000000
0000000000111111111111111111000111110000000000000000
0000001011111111111111111111010101100001000000000000
0000000011111111111111111111000111111100000000000000
0000011101111111111111111111001111101111000000000000
0000000111111111111011111110011111111111000000000000
0000000111011111111111111111001111111000000000000000
0000000111011111011111111111001111110000000000000000
0000000011111111111011111111000111110000000000000000
0000000001111111100111111111000011100000000000000000
0000000000111111111111111111100110000000000000000000
0000000000111111111111111111000000000000000000000000
0000000000011111111111111111000000000000000000000000
0000000000011111111111111111000000000000000000000000
0000000000001111101111111100000000000000000000000000
0000000000000000111000101000100000000000000000000000
0000000000000000011011111101000000000000000000000000
0000000000000000001111001110000000000000000000000000
0000000000000000000111111000000000000000000000000000
0000000000000000000011011000000000000000000000000000
0000000000000000000001110000000000000000000000000000
0000000000000000000000110000000000000000000000000000
0000000000000000000000010000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
```

CPrettyPrint File:

```
46 46 0 1
```