Student: Michael Grossman

Project Due Date: 3/17/2022

Algorithm Steps for ComputeGauss given a frameArray, maskArray, gaussArray, and a weight:

0. newMin←9999 ; newMax ←0
1. i←2
2. j←2
3. gaussArray[i,j]←(int)(convolution5x5(i,j,frameArray,maskArray) / weight)
4. if newMin > gaussArray[i,j]
5. newMin ← gaussArray[i,j]
6. end-if
7. if newMax < gaussArray[i,j]
8. newMax ← gaussArray[i,j]
9. end-if
10. j++
11. repeat 3 to 10 while j < numCols + 2
12. i++
13. repeate 3 to 12 while i < numRows + 2

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

class imageProcessing{
    public:
        int numRows, numCols, minVal, maxVal;
        int newMin, newMax;
        int maskRows, maskCols, maskMin, maskMax;
        int thr;
        int **frameArray, **gaussArray, **thrArray, **maskArray;
        int weight;

        imageProcessing(int* imH, int* mskH, int t);
        ~imageProcessing();
        void loadImage(ifstream& in, int** ary);
        void mirrorFraming();
        int loadMask(ifstream& in);
        int convolutions5x5(int i, int j, int** fArray, int** mArray);
        void computeGaus(int** fAarray, int** mArray, int** gArray, int w);
        void threshold(int** ary, int** tAry, int t);
        void imageReformat(int** inAry, int newMin, int newMax, ofstream&
output);
        void prettyPrint(int** ary, ofstream& out);
};

int main(int argc, char** argv){
    string inputFileName = argv[1], maskFileName = argv[2], outputFileName =
argv[4];
    int threshold = atoi(argv[3]);

    ifstream input(inputFileName), mask(maskFileName);
    ofstream output(outputFileName);
    int inputHeader[4], maskHeader[4];
    for(int i = 0; i < 4; ++i){
        input >> inputHeader[i];
        mask >> maskHeader[i];
    }
    imageProcessing imageprocessing(inputHeader, maskHeader, threshold);
    imageprocessing.loadImage(input, imageprocessing.frameArray);
    imageprocessing.weight = imageprocessing.loadMask(mask);
    imageprocessing.imageReformat(imageprocessing.frameArray,
imageprocessing.minVal, imageprocessing.maxVal, output);
```

```
    imageprocessing.computeGaus(imageprocessing.frameArray,
imageprocessing.maskArray, imageprocessing.gaussArray, imageprocessing.weight);
    imageprocessing.imageReformat(imageprocessing.gaussArray,
imageprocessing.newMin, imageprocessing.newMax, output);
    imageprocessing.threshold(imageprocessing.gaussArray,
imageprocessing.thrArray, imageprocessing.thr);
    imageprocessing.imageReformat(imageprocessing.thrArray,
imageprocessing.newMin, imageprocessing.newMax, output);
    imageprocessing.prettyPrint(imageprocessing.thrArray, output);


    input.close();
    output.close();
    mask.close();
}

imageProcessing::imageProcessing(int* imH, int* mskH, int t){
    numRows = imH[0];
    numCols = imH[1];
    minVal = imH[2];
    maxVal = imH[3];
    maskRows = mskH[0];
    maskCols = mskH[1];
    maskMin = mskH[2];
    maskMax = mskH[3];
    thr = t;

    frameArray = new int*[numRows + 4];
    gaussArray = new int*[numRows + 4];
    thrArray = new int*[numRows + 4];
    maskArray = new int*[maskRows];

    for(int i = 0; i < numRows + 4; ++i){
        frameArray[i] = new int[numCols + 4]{0};
        gaussArray[i] = new int[numCols + 4]{0};
        thrArray[i] = new int[numCols + 4]{0};
    }
    for(int i = 0; i < maskRows; ++i){
        maskArray[i] = new int[maskCols]{0};
    }

}

imageProcessing::~imageProcessing(){
    int frameSizeRows = numRows + 4;
```

```
    for(int i = 0; i < frameSizeRows; ++i){
        delete[] frameArray[i];
        delete[] gaussArray[i];
        delete[] thrArray[i];
    }

    delete[] frameArray;
    delete[] gaussArray;
    delete[] thrArray;

    for(int i = 0; i < maskRows; ++i){
        delete[] maskArray[i];
    }
    delete[] maskArray;
}

void imageProcessing::loadImage(ifstream& in, int** ary){
    int rows = numRows+2, cols = numCols + 2;
    for(int i = 2; i < rows; ++i){
        for(int j = 2; j < cols; ++j){
            in >> frameArray[i][j];
        }
    }
    mirrorFraming();
}


void imageProcessing::mirrorFraming(){
    int frameRows = numRows + 4, frameCols = numCols + 4;

    //mirror top then bottom
    for(int i = 0; i < 2; ++i){
        for(int j = 2; j < numCols+2; ++j){
            frameArray[i][j] = frameArray[3-i][j];
        }
    }
    for(int i = frameRows- 2; i < frameRows; ++i){
        for(int j = 2; j < numCols + 2; ++j){
            frameArray[i][j] = frameArray[2*frameRows-5 - i][j];
        }
    }

    //mirror left then right
    for(int i = 2; i < frameRows-2; ++i){
```

```
        for(int j = 0; j < 2; ++j){
            frameArray[i][j] = frameArray[i][3-j];
        }
    }
    for(int i = 2; i < frameRows - 2; ++i){
        for(int j = frameCols-2; j < frameCols; ++j){
            frameArray[i][j] = frameArray[i][2*frameCols-5-j];
        }
    }

    //mirror corners, reflected over appropriate corner
    frameArray[0][0] = frameArray[3][3];
    frameArray[1][1] = frameArray[2][2];
    frameArray[0][1] = frameArray[2][3];
    frameArray[1][0] = frameArray[3][2];
    frameArray[0][frameCols-2] = frameArray[2][frameCols-4];
    frameArray[0][frameCols-1] = frameArray[3][frameCols-4];
    frameArray[1][frameCols-2] = frameArray[2][frameCols-3];
    frameArray[1][frameCols-1] = frameArray[3][frameCols-3];
    frameArray[frameRows-2][0] = frameArray[frameRows-4][2];
    frameArray[frameRows-2][1] = frameArray[frameRows-3][2];
    frameArray[frameRows-1][0] = frameArray[frameRows-4][3];
    frameArray[frameRows-1][1] = frameArray[frameRows-3][3];
    frameArray[frameRows-2][frameCols-2] = frameArray[frameRows-3][frameCols-3];
    frameArray[frameRows-2][frameCols-1] = frameArray[frameRows-4][frameCols-3];
    frameArray[frameRows-1][frameCols-2] = frameArray[frameRows-3][frameCols-4];
    frameArray[frameRows-1][frameCols-1] = frameArray[frameRows-4][frameCols-4];
}


int imageProcessing::loadMask(ifstream& in){
    int w = 0;
    for(int i = 0; i < maskRows; ++i){
        for(int j = 0; j < maskCols; ++j){
            in >> maskArray[i][j];
            w += maskArray[i][j];
        }
    }
    return w;
}


int imageProcessing::convolutions5x5(int i, int j, int** fArray, int** mArray){
    int retVal = 0;
    for(int rows = i-2; rows <= i + 2; ++rows){
```

```
        for(int cols = j-2; cols <= j+2; ++cols){
            retVal += mArray[rows-i+2][cols-j+2]*fArray[rows][cols];
        }
    }
    return retVal;
}


void imageProcessing::computeGaus(int** fAarray, int** mArray, int** gArray, int
w){
    newMin = 99999;
    newMax = 0;
    for(int i = 2; i < numRows + 2; ++i){
        for(int j = 2; j< numCols + 2; ++j){
            gArray[i][j] = (int) (convolutions5x5(i, j, fAarray, mArray) / w);
            newMin = newMin > gArray[i][j] ? gArray[i][j] : newMin;
            newMax = newMax < gArray[i][j] ? gArray[i][j] : newMax;
        }
    }
}


void imageProcessing::threshold(int** ary, int** tAry, int t){
    for(int i = 0; i < numRows+4; ++i){
        for(int j = 0; j < numCols + 4; ++j){
            tAry[i][j] = ary[i][j] >= t ? 1 : 0;
        }
    }
}


void imageProcessing::imageReformat(int** inAry, int newMin, int newMax,
ofstream& output){
    output << numRows << " ";
    output << numCols << " ";
    output << newMin << " ";
    output << newMax << "\n";
    string str = to_string(newMax);
    int width = str.length();
    int r =0, c =0, ww = 0;

    for(int r = 0; r < numRows+4; ++r){
        for(int c = 0; c < numCols+4; ++c){
            output << inAry[r][c];
            str = to_string(inAry[r][c]);
```

```
            output << " ";
            for(ww = str.length(); ww < width; ++ww){
                output << " ";
            }
        }
        output << "\n";
    }
    output << "\n\n";
}


void imageProcessing::prettyPrint(int** ary, ofstream& out){
    for(int i = 2; i < numRows + 2;++i){
        for(int j = 2; j < numCols + 2; ++j){
            if(ary[i][j] > 0){
                out << "1 ";
            }
            else{
                out << ". ";
            }
        }
        out << "\n";
    }
    out << "\n\n";
}
```

```
46 46 1 63
1 1 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 4 4
2 1 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 1 2 3 4 5 5 5
1 1 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 1 2 3 4 5 5 4
1 2 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 5 4
1 3 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 5 4
1 4 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 4
1 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 5 4
1 6 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 5 4
1 7 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 5 4
1 8 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 5 4
1 9 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 5 4
11 10 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 5 4
1 1 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 5 4
1 2 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 5 4
1 3 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 5 4
21 4 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 14
1 5 5 1 21 3 4 5 1 2 3 4 5 1 2 3 4 5 1 48 48 48 58 10 41 48 42 34 48 45 48 48 5 1 2 3 4 5 1 2 3 4 5 1 32 3 4 5 5 4
1 6 6 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 48 48 58 48 58 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 5 4
1 7 7 1 2 3 14 5 1 2 3 4 5 1 2 3 4 48 48 38 41 42 43 41 42 63 4 48 48 46 48 48 48 48 2 3 4 51 1 2 3 4 5 1 2 3 4 5 5 4
1 8 8 1 2 3 4 5 1 2 3 4 5 1 2 3 48 41 38 44 43 48 45 48 44 48 48 48 48 58 48 48 4 4 48 3 4 5 1 2 3 4 5 1 2 3 4 5 5 4
1 9 9 1 2 3 4 15 1 12 3 4 5 1 2 48 48 60 48 60 41 48 41 48 48 61 62 48 48 43 48 8 7 48 48 48 4 5 1 2 3 4 5 1 2 13 4 5 5 4
1 10 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 5 4
1 1 1 1 52 3 4 5 1 12 3 4 5 48 41 58 48 43 48 48 48 48 28 38 48 43 48 48 41 48 8 48 45 28 48 42 48 48 1 2 3 4 5 11 2 3 4 5 5 4
1 2 2 1 2 3 4 5 1 2 3 4 48 48 58 48 48 48 40 48 47 48 42 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 5 14
21 3 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 5 14
1 4 4 1 2 3 4 5 1 2 48 48 48 48 48 48 4 58 48 48 38 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 4
1 5 5 1 2 48 41 48 42 48 43 8 48 60 58 48 48 38 41 42 48 43 48 46 48 45 48 40 48 4 3 48 30 48 48 48 8 42 48 48 38 48 48 28 48 48 4 5 5 4
1 6 6 1 2 3 4 5 1 2 48 58 58 48 38 48 48 38 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 5 4
1 7 7 1 2 3 4 5 1 2 3 48 48 8 48 38 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 5 4
1 8 8 1 2 3 4 5 13 2 3 4 48 48 62 63 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 5 4
1 9 9 1 2 3 4 5 1 2 3 4 5 48 48 58 48 38 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 5 4
1 10 10 1 2 3 4 5 1 2 3 4 5 1 48 48 58 48 3 48 58 48 40 48 58 18 48 48 48 41 48 42 48 8 4 48 48 5 1 2 3 4 5 1 12 3 4 5 5 4
21 1 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 5 14
1 2 2 1 2 3 4 5 1 2 3 4 5 1 2 48 63 48 48 48 18 48 42 48 41 48 44 48 42 48 8 48 48 8 4 48 4 5 1 2 3 14 15 1 2 3 4 5 5 4
11 3 3 11 2 3 4 5 1 2 3 4 5 1 2 3 48 42 48 43 48 48 48 48 58 48 48 58 48 58 48 48 48 8 4 3 4 5 1 2 3 4 5 1 2 3 4 5 5 4
1 4 4 1 2 3 4 15 1 2 3 4 5 1 2 3 4 48 48 41 42 63 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 4
1 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 5 4
1 6 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 5 4
1 7 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 5 4
1 8 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 5 4
1 9 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 5 4
1 10 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 5 4
1 1 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 5 4
1 2 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 5 4
1 3 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 5 4
1 4 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 4
1 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 5 4
1 6 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 5 4
5 6 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 5 5
1 1 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 4 4
```

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


46 46 1 51