Name: Alif Rahi
Section: CSCI 381 - Computer Vision / Tues-Thurs 1:40-2:55pm
Project: 2
Due: Feb 19, 2023

Main algorithm steps:

Step 0: open inFile, maskFile via argv[] open imgOutFile, AvgOutFile, MedianOutFile, GaussOutFile via argv[] thrValget from argv[3]

Step 1: numRows, numCols, minVal, maxValread from inFile maskRows, maskCols, maskMin, maskMaxread from maskFile

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

Step 3: loadMaskAry (maskFile, maskAry)

Step 4: loadImage (inFile, mirrorFramedAry)

Step 5: mirrorFraming (mirrorFramedAry)

Step 6: imgReformat (mirrorFramedAry, deBugFile)

Source code

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

class Enhancement
{

public:
   int rows, cols, minVal, maxVal, maskRows, maskCols, maskMinVal, maskMaxVal,
thrVal;
   int **mirrorFramedAry, **avgAry, **medianAry, **GaussAry, **thrAry,
*neighborAry, *maskAry, maskWeight;

   Enhancement(int r, int c, int mnV, int mxV, int mR, int mC, int mMin, int
mMax, int tVal)
   {
       rows = r;
       cols = c;
       minVal = mnV;
       maxVal = mxV;
       maskRows = mR;
       maskCols = mC;
       maskMinVal = mMin;
```

```
    maskMaxVal = mMax;
    thrVal = tVal;
}

void initArrays(int numRows, int numCols)
{

    mirrorFramedAry = new int *[numRows + 4];
    avgAry = new int *[numRows + 4];
    medianAry = new int *[numRows + 4];
    GaussAry = new int *[numRows + 4];
    thrAry = new int *[numRows + 4];

    for (int i = 0; i < numRows + 4; ++i)
    {
        mirrorFramedAry[i] = new int[numCols + 4];
        avgAry[i] = new int[numCols + 4];
        medianAry[i] = new int[numCols + 4];
        GaussAry[i] = new int[numCols + 4];
        thrAry[i] = new int[numCols + 4];
    }

    neighborAry = new int[25];
    maskAry = new int[25];
};

void loadImage(ifstream &inFile, int r, int c)
{
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            int val;
            inFile >> val;
            mirrorFramedAry[i + 2][j + 2] = val;
        }
    }
}

void mirrorFraming(int iR, int iC)
{
```

```
    mirrorFramedAry[iR][iC] = mirrorFramedAry[iR + 1][iC + 1];
    int i;
    for (i = iR + 1; i < rows + 4 - iR; i++)
    {
        mirrorFramedAry[i][iC] = mirrorFramedAry[i][iC + 1];
    }
    i--;
    mirrorFramedAry[i][iC] = mirrorFramedAry[i - 1][iC + 1];

    int j;
    for (j = iC; j < cols + 4 - iC; j++)
    {
        mirrorFramedAry[i][j] = mirrorFramedAry[i - 1][j];
    }

    j--;
    mirrorFramedAry[i][j] = mirrorFramedAry[i - 1][j - 1];
    int k;
    for (k = i; k >= iR; k--)
    {
        mirrorFramedAry[k][j] = mirrorFramedAry[k][j - 1];
    }

    mirrorFramedAry[iR][j] = mirrorFramedAry[iR + 1][j - 1];
    for (int j = i; j >= iC; j--)
    {
        mirrorFramedAry[iR][j] = mirrorFramedAry[iR + 1][j];
    }
}

void loadMaskAry(ifstream &maskFile)
{
    int sum = 0;
    for (int i = 0; i < 25; i++)
    {
        int val;
        maskFile >> val;
        maskAry[i] = val;
        sum += val;
    }
    maskWeight = sum;
};
```

```
void loadNeighborAry(int i, int j)
{
    int idx = 0;
    for (int k = i - 2; k <= i + 2; k++)
    {
        for (int l = j - 2; l <= j + 2; l++)
        {
            neighborAry[idx++] = mirrorFramedAry[k][l];
            cout << mirrorFramedAry[k][l] << " ";
        }
    }
    cout << endl;
}

void sortArr()
{
    int temp;
    for (int i = 0; i < 25; i++)
    {
        for (int j = i + 1; j < 25; j++)
        {
            if (neighborAry[j] < neighborAry[i])
            {
                temp = neighborAry[i];
                neighborAry[i] = neighborAry[j];
                neighborAry[j] = temp;
            }
        }
    }
}

void computeAvg()
{
    for (int i = 2; i < rows + 2; i++)
    {
        for (int j = 2; j < cols + 2; j++)
        {
            loadNeighborAry(i, j);
            int sum = 0;
            for (int k = 0; k < 25; k++)
            {
```

```
                sum += neighborAry[k];
            }

            avgAry[i][j] = sum / 25;
        }
    }
}

void
computeMedian()
{
    for (int i = 2; i < rows + 2; i++)
    {
        for (int j = 2; j < cols + 2; j++)
        {
            loadNeighborAry(i, j);
            sortArr();
            medianAry[i][j] = neighborAry[12];
        }
    }
}

int convolution()
{
    int result = 0;
    for (int i = 0; i < 25; i++)
    {
        result += neighborAry[i] * maskAry[i];
    }

    return result / maskWeight;
}

void computeGauss()
{
    for (int i = 2; i < rows + 2; i++)
    {
        for (int j = 2; j < cols + 2; j++)
        {
            loadNeighborAry(i, j);
            GaussAry[i][j] = convolution();
        }
```

```
        }
    }

    void imgReformat(int **arr, ofstream &debugFile, string msg)
    {
        debugFile << msg << endl;
        debugFile << rows << " " << cols << " " << minVal << " " << maxVal <<
endl;
        string str = to_string(maxVal);
        int width = str.length();
        for (int r = 2; r < rows + 2; r++)
        {
            for (int c = 2; c < cols + 2; c++)
            {
                debugFile << arr[r][c] << " ";
                str = to_string(arr[r][c]);
                int WW = str.length();
                while (WW < width)
                {
                    debugFile << " ";
                    WW++;
                }
            }
            debugFile << endl;
        }
        debugFile << endl;
    }

    void binaryThreshold(int **arr)
    {
        for (int i = 0; i < rows + 4; i++)
        {
            for (int j = 0; j < cols + 4; j++)
            {
                if (arr[i][j] >= thrVal)
                {
                    thrAry[i][j] = 1;
                }
                else
                {
                    thrAry[i][j] = 0;
                }
```

```
            }
        }
    }

    void prettyPrint(ofstream &outFile)
    {
        for (int i = 0; i < rows + 4; i++)
        {
            for (int j = 0; j < cols + 4; j++)
            {
                if (thrAry[i][j] > 0)
                {
                    outFile << thrAry[i][j] << " ";
                }
                else
                {
                    outFile << "  ";
                }
            }
            outFile << endl;
        }
    }
};

int main(int argc, char **argv)
{
    ifstream inFile(argv[1]);
    int rows, cols, minVal, maxVal, maskRows, maskCols, maskMinVal, maskMaxVal;
    inFile >> rows >> cols >> minVal >> maxVal;
    ifstream maskFile(argv[2]);
    maskFile >> maskRows >> maskCols >> maskMinVal >> maskMaxVal;
    int threshold = stoi(argv[3]);
    ofstream imgOutFile(argv[4]);
    ofstream AvgOutFile(argv[5]);
    ofstream MedianOutFile(argv[6]);
    ofstream GaussOutFile(argv[7]);
    ofstream debugFile(argv[8]);

    Enhancement enhancement(rows, cols, minVal, maxVal, maskRows, maskCols,
maskMinVal, maskMaxVal, threshold);
    enhancement.initArrays(rows, cols);
```

```
    enhancement.loadImage(inFile, rows, cols);
    enhancement.mirrorFraming(1, 1);
    enhancement.mirrorFraming(0, 0);
    enhancement.loadMaskAry(maskFile);

    // step 7
    enhancement.computeAvg();
    enhancement.imgReformat(enhancement.avgAry, debugFile, "Avg filtering");
    enhancement.binaryThreshold(enhancement.avgAry);
    enhancement.prettyPrint(AvgOutFile);

    // step 8
    enhancement.computeMedian();
    enhancement.imgReformat(enhancement.medianAry, debugFile, "Median
filtering");
    enhancement.binaryThreshold(enhancement.medianAry);
    enhancement.prettyPrint(MedianOutFile);

    // step 9
    enhancement.computeGauss();
    enhancement.imgReformat(enhancement.GaussAry, debugFile, "Gaussian
filtering");
    enhancement.binaryThreshold(enhancement.GaussAry);
    enhancement.prettyPrint(GaussOutFile);

    inFile.close();
    maskFile.close();
    imgOutFile.close();
    AvgOutFile.close();
    MedianOutFile.close();
    GaussOutFile.close();
    debugFile.close();

    return 0;
}
```

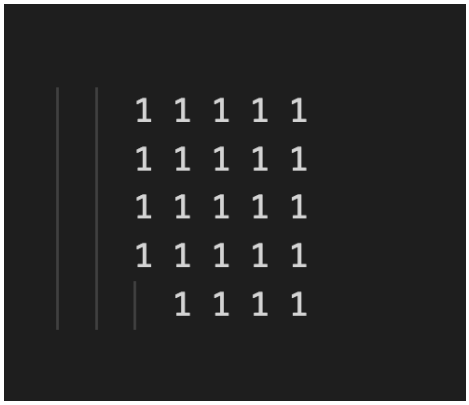
Output

img1:

MedianOutputFile

GaussianOutputFile

AvgOutputFile



Debug file

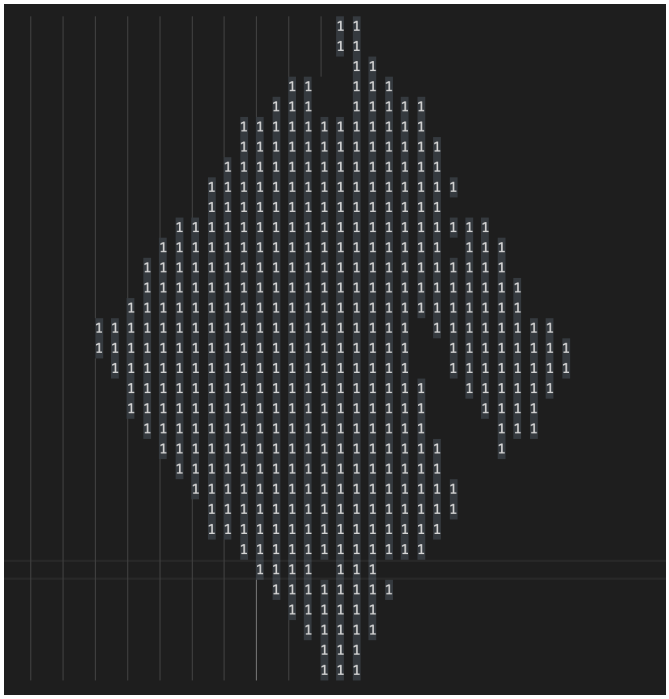
```
Avg filtering
5 5 1 36
8  10 10 10 9
8  9  9  9  8
7  9  8  8  8
5  8  8  8  7
4  6  6  6  6

Median filtering
5 5 1 36
5  5  5  4  4
5  5  5  4  4
5  5  5  4  4
4  5  4  4  4
2  2  4  4  4

Gaussian filtering
5 5 1 36
7  9  12 10 7
7  11 12 12 8
7  11 13 12 8
5  7  8  8  6
3  4  5  5  4
```

Output

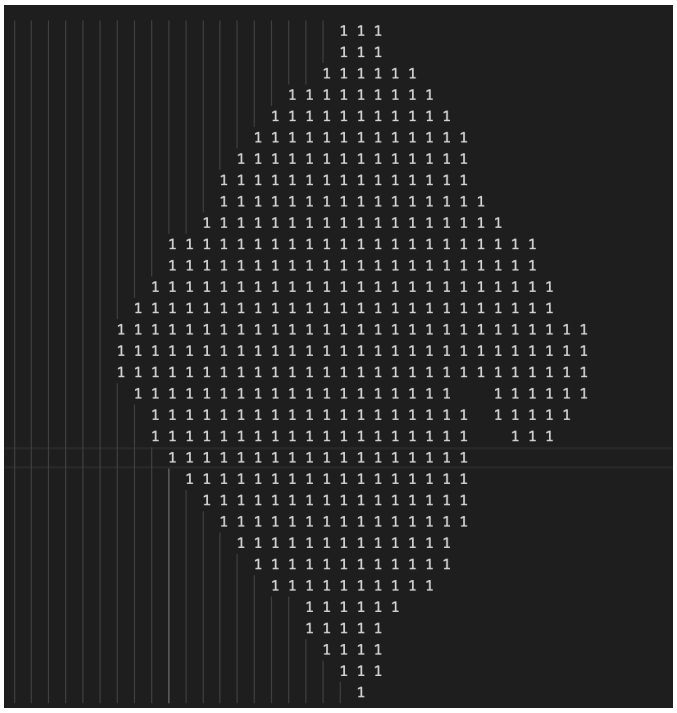
img2:

MedianOutputFile

GaussianOutputFile



AvgOutputFile



Debug file


Avg filtering
46 46 1 63
Median filtering
46 46 1 63
Gaussian filtering
46 46 1 63