Student: Michael Grossman

Project Due Date: 02/13/2022

Algorithm Steps for Computing the Histogram Given an Input-File-Stream:

1. For Each Pixel in the file-stream:
2. Val ←Pixel
3. histArray[Val]++
4. End-Loop

Algorithm Steps for Creating the Threshold Images Given a Threshold-Value, an Input File-Stream, and Two Output File-Streams:

1. minVal ← 0
2. maxVal ← 1
3. Output-File-Stream1, Output-File-Stream2 ← output numRows, numCols, minVal, and maxVal
4. For Each Pixel in the Input-File-Stream:
5. PixelValue ←Pixel
6. If PixelValue >= Threshold-Value:
7. Output-File-Stream1, Output-File-Stream2 ← “1 “
8. Else:
9. Output-File-Stream1 ← “0 “
10. Output-File-Stream2 ← “. “
11. End If-Else
12. Output-File-Stream1, Output-File-Stream2 ← “\n”
13. End-Loop

**Main.cpp**

#include <iostream>

#include <fstream>

/\*

Program Specs:

1. Compute histogram of the input image and display the histogram in two

formats, see the output description below.

2. Perform binary threshold operation on the input image with a given

threshold value via argv[].

3. Output the result of the threshold in two formats, see the output

description below

\*/

class Image{

    public:

        //constructor

        Image(int nr, int nc, int mnv, int mxv, int tv);

        //destructor

        ~Image();

        //attributes

        int numRows;

        int numCols;

        int minVal;

        int maxVal;

        int\* histArray;

        int thresholdVal;

        //functions

        void computeHist(std::ifstream& input);

        void printHist(std::ofstream& output);

        void dispHist(std::ofstream& output);

        void threshold(std::ifstream& input,

                        std::ofstream& output1,

                        std::ofstream& output2, int tv);

};

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

    if(argc > 7 || argc < 7){

        std::cout << "incorrect command line args - must always"

                    "be in the format: " << std::endl;

        std::cout << "./program file\_name threshold\_value "

                    "output\_file1 output\_file2 output\_file3 "

                    "output\_file\_4" << std::endl;

        return 0;

    }

    //store command line args

    std::string in\_file\_name = argv[1];

    int threshold\_value = atoi(argv[2]);

    std::string out\_file\_names[4] = {argv[3], argv[4], argv[5], argv[6]};

    //open the files

    std::ifstream fin(in\_file\_name);

    std::ofstream fout1(out\_file\_names[0]), fout2(out\_file\_names[1]),

                    fout3(out\_file\_names[2]), fout4(out\_file\_names[3]);

    //get the image's header info

    int image\_header\_info[4];

    for(int i = 0; i < 4; ++i){

        fin >> image\_header\_info[i];

    }

    //apparently not allowed to use the getline / stringstream method,

    //leaving here in case of errors

    /\*

    std::string current\_line;

    std::getline(fin, current\_line);

    int image\_header\_info[4];

    std::string working\_string;

    std::stringstream sstream(current\_line);

    for(int i = 0; i < 4; ++i){

        std::getline(sstream, working\_string, ' ');

        image\_header\_info[i] = stoi(working\_string);

    }

    \*/

    //create image object

    Image image(image\_header\_info[0], image\_header\_info[1],

            image\_header\_info[2], image\_header\_info[3],

            threshold\_value);

    //compute the histogram and print out to files

    image.computeHist(fin);

    image.printHist(fout1);

    image.dispHist(fout2);

    fin.close();

    //reopen input file

    fin.open(in\_file\_name);

    //output new thresholded image

    fout3 << "The threshold value uses is " << threshold\_value << "\n";

    fout4 << "The threshold value uses is " << threshold\_value << "\n";

    image.threshold(fin, fout3, fout4, threshold\_value);

    //close all streams

    fin.close();

    fout1.close();

    fout2.close();

    fout3.close();

    fout4.close();

    return 0;

}

Image::Image(int nr, int nc, int mnv, int mxv, int tv) : numRows(nr),

                numCols(nc), minVal(mnv), maxVal(mxv), thresholdVal(tv)

{

    int arr\_end = mxv + 1;

    histArray = new int[arr\_end];

    for(int i = 0; i < arr\_end; ++i){

        histArray[i] = 0;

    }

}

Image::~Image(){

    delete[] histArray;

}

void Image::computeHist(std::ifstream& input){

    int total\_pixels = numRows \* numCols, working\_int = 0;

    for(int i = 0; i < total\_pixels; ++i){

        input >> working\_int;

        histArray[working\_int]++;

    }

    //apparently we cannot use the getline / stringstream method,

    //leaving here in case of errors

    /\*

    std::string working\_str;

    std::stringstream working\_stream;

    while(!input.eof()){

        std::getline(input, working\_str);

        working\_stream = std::stringstream(working\_str);

        while(!working\_stream.eof()){

            std::getline(working\_stream, working\_str, ' ');

            if(working\_str[0] < '0' || working\_str[0] > '9') break;

            histArray[std::stoi(working\_str)]++;

        }

    }

    \*/

}

void Image::printHist(std::ofstream& output){

    output << numRows << " " << numCols << " " << minVal

            << " " << maxVal << "\n";

    int arr\_end = maxVal + 1;

    for(int i = 0; i < arr\_end; ++i){

        output << i << " " << histArray[i] << "\n";

    }

}

void Image::dispHist(std::ofstream& output){

    output << numRows << " " << numCols << " " << minVal

             << " " << maxVal << "\n";

    int arr\_end = maxVal + 1;

    int max = 0;

    for(int i = 0; i < arr\_end; ++i){

        output << i << " (" << histArray[i] << "):";

        max = histArray[i] > 70 ? 70 : histArray[i];

        for(int j = 0; j < max; ++j){

            output << "+";

        }

        output << "\n";

    }

}

void Image::threshold(std::ifstream& input, std::ofstream& output1,

                        std::ofstream& output2, int tv){

    minVal = 0;

    maxVal = 1;

    int pixelVal = 0;

    output1 << numRows << " " << numCols << " " << minVal

            << " " << maxVal << "\n";

    output2 << numRows << " " << numCols << " " << minVal

            << " " << maxVal << "\n";

    //move passed the header

    for(int i = 0; i < 4; ++i){

        input >> pixelVal;

    }

    //read in the file and output depending on whether the value

    //meets the given threshold tv

    for(int i = 0; i < numRows; ++i){

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

            input >> pixelVal;

            if(pixelVal >= tv){

                output1 << "1 ";

                output2 << "1 ";

            }

            else{

                output1 << "0 ";

                output2 << ". ";

            }

        }

        output1 << "\n";

        output2 << "\n";

    }

    /\* apparently we cant use the getline/stringstream method,

    //leaving here in case of errors

    std::string working\_str;

    std::stringstream working\_stream;

    std::getline(input, working\_str);

    while(!input.eof()){

        std::getline(input, working\_str);

        working\_stream = std::stringstream(working\_str);

        while(!working\_stream.eof()){

            std::getline(working\_stream, working\_str, ' ');

            if(working\_str[0] < '0' || working\_str[0] > '9') break;

            pixelVal = std::stoi(working\_str);

            if(pixelVal >= tv){

                output1 << "1 ";

                output2 << "1 ";

            }

            else{

                output1 << "0 ";

                output2 << ". ";

            }

        }

        output1 << "\n";

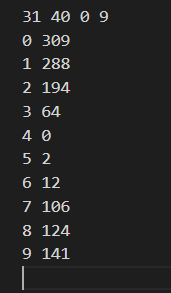
        output2 << "\n";

    }

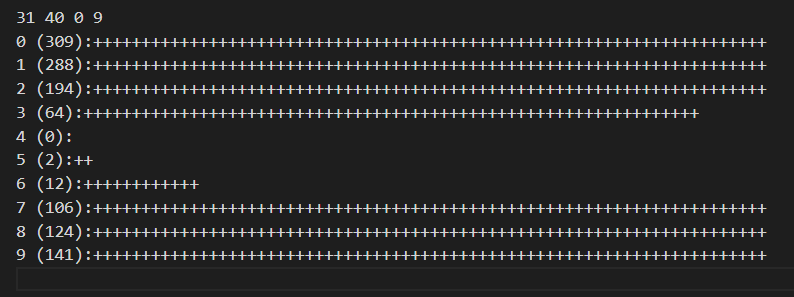
    \*/

}

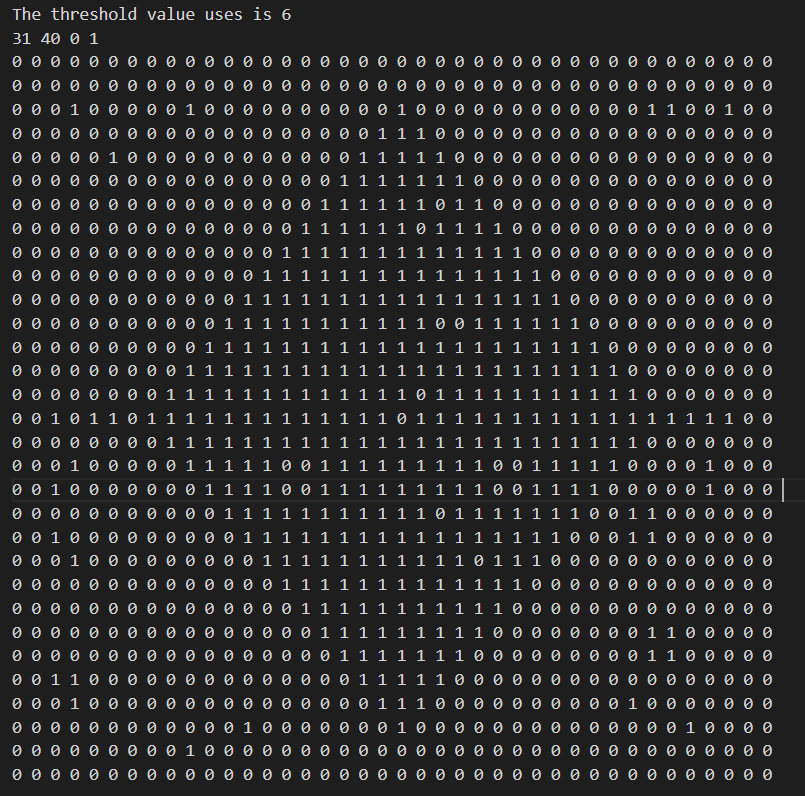
**Output outFile1 for data 1**



**Output outFile2 for data 1**



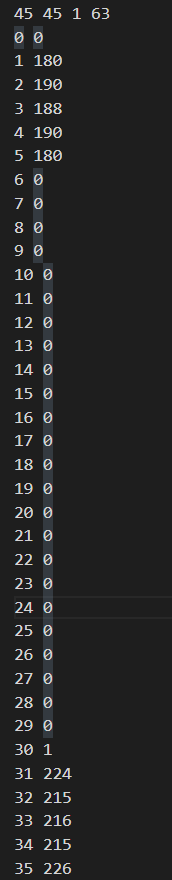
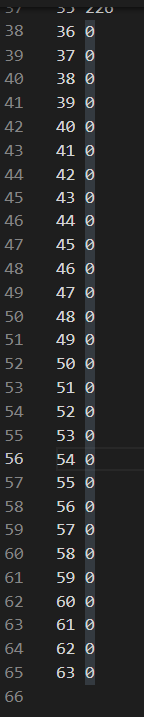
**Output outFile3 for data 1**

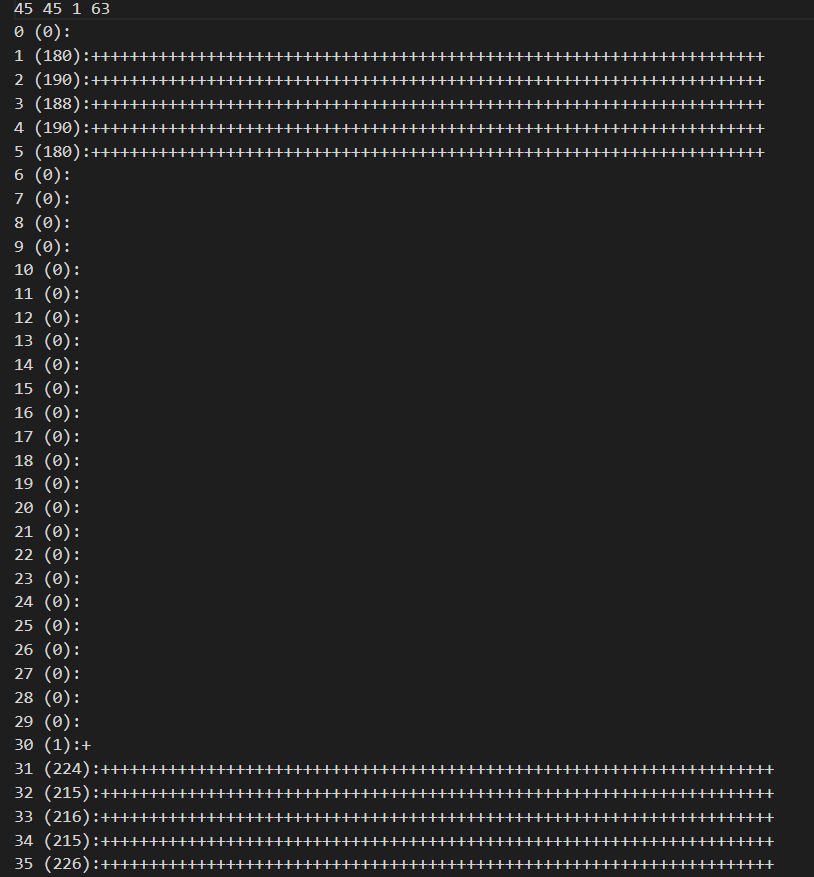
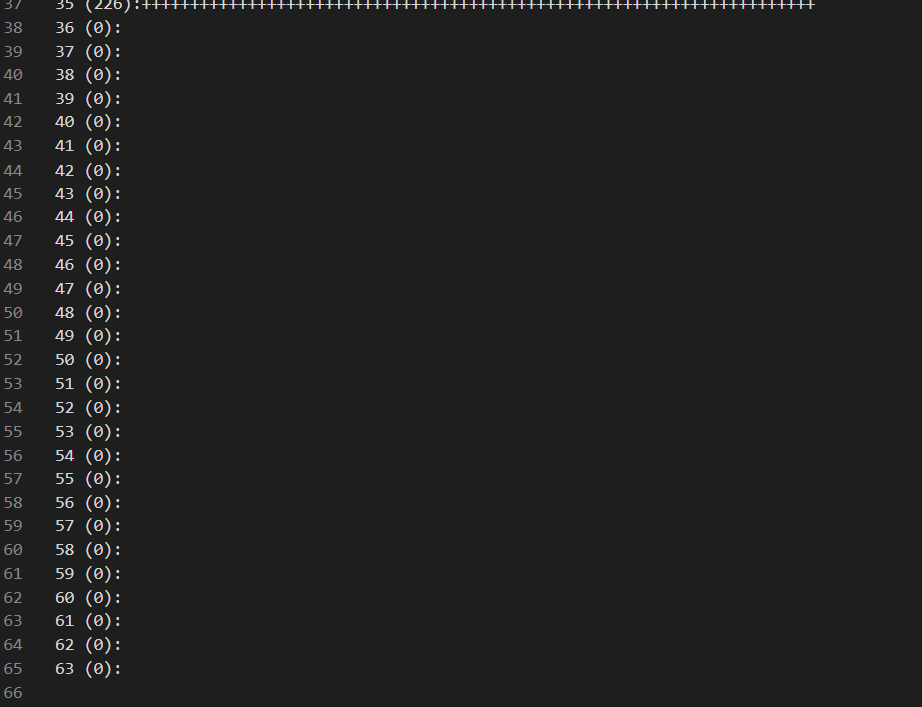
****

**Output outFile4 for data 1**



**Output outFile1 for data 2**

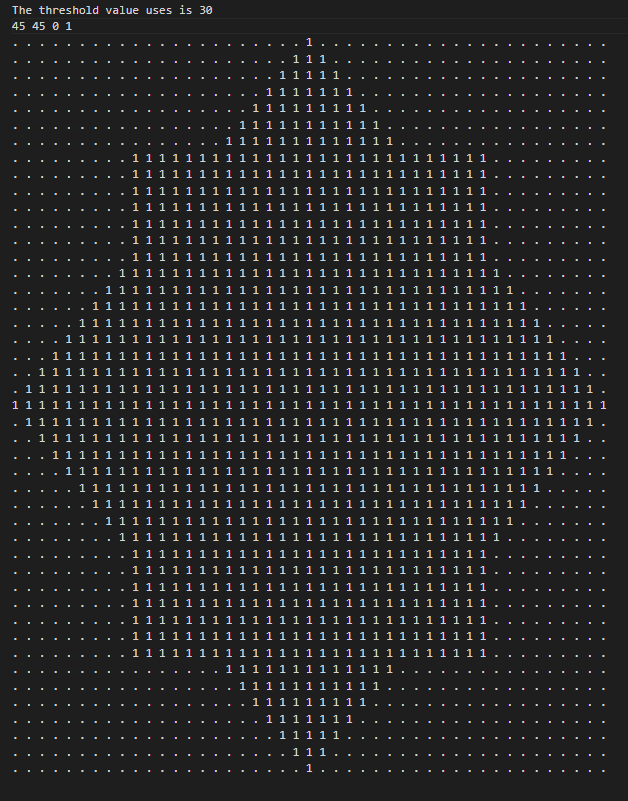


**Output outFile2 for data 2**continued on next page

**Output outFile3 for data 2**

****

**Output outFile4 for data 2**

****