Project 1 (in C++): Histogram and thresholding. Given a grey-scale image, write a program to perform the following tasks:

a) Pretty print the input image.
b) Compute histogram of the input image.
c) Output the histogram in two formats, see the output description below.
d) Perform binary threshold operation on the input image with a given threshold value via argv [2].
e) Output the result of binary threshold operation in the format given in the output description below.

What you need to do:

1. Implement your program with respect to the specs given below and debug your program until your program compiles.
2. You will be given two data files: data1 and data2.
3. Run your program twice: once using data1 with threshold 6 (via argv[2]) and once using data2 with threshold 29.
4. Before you submit, look at your outputs and pay attention to see if the image you display are line-up pixel by pixel, and fit within the width of the page. (Use “Courier New” font and small font size, 4-6.)

*** Include in your hard copy *PDF.pdf file as follows:

- Cover page.
- source code.
- Output histCountFile for data1.
- Output histGraphFile for data1.
- Output binThrFile for data1.
- Output logFile for data1.
- Output histCountFile for data2.
- Output histGraphFile for data2.
- Output binThrFile for data2.
- Output logFile for data2.

*************************************

Language: C++

*************************************

Project#: Project 1

Project name: Histogram and thresholding

Project points: 10 pts

Due Date: Soft copy (*.zip) and hard copies (*.pdf):

+1 (11/10 pts): early submission, 9/8/2024, Sunday before midnight (11:59pm)
(10/10 pts): on time, 9/11/2024. Wednesday before midnight
(-10/10 pts): non-submission, 9/11/2024. Wednesday after midnight

*** Name your soft copy and hard copy files using the naming convention given in Project Submission Requirements.

***** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in the same email attachments with correct email subject as stated in the email requirement; otherwise, your submission will be rejected.**

********************************************

I. Input: There are two inputs to the program via argv[1] and argv[2].

********************************************

a) inFile (argv[1]): a txt file representing a grey-scale image, where the first text line (4 integers) is the "header" of the input image then follows by rows and cols of integers.

For example,

```
4  6  1 12          // image has 4 rows,6 cols, min is 1, max is 12
2  3  4  11 2  9
5  6 11  2 10 7
1  1 12  1  9  9
4  5   6  9  9  9
```

**b) a threshold value (argv[2]): use 6 for data1 and 29 for data2.**

******************************************
II.     Outputs: There are 4 output files.
******************************************

a) histCountFile (use argv[3]): For the output of histogram in the following format:
The first text-line is the image header, follows by a list of pairs <i, j> where i = 0 to max and j is the counts of pixels having i value.  For example:

```
4 6 1 12   // image header.
0   0
1   3
2   3
3   1
4   2
5   2
6   2
7   1
8   0
9   6
10  1
11  2
12  1
```

b) histGraphFile (use argv[4]): Display the histogram (for visual).  Use the following format:
The first text line is the image header then follows by a list of  greyScale (pixel count): number of +'s;
Choose a small font size so that each bin can fit in one text-line).

For example

```
4   6   1   12     // image header.
0  (0):
1  (3):+++
2  (3):+++
3  (1):+
4  (2):++
5  (2):++
6  (2):++
7  (1):+
8  (0):
9  (6):++++++
10 (1):+
11 (2):++
12 (1):+
```

c) binThrFile (use argv[5]): The result of the threshold operation on the input image, in the format given below. (**Use Courier New font with small font size so your output image will fit within a page and pixels are line-up nicely. (-1 pt for not doing so.)**

*** The result of the binary image using 6 as threshold value ***

```
4   6   0   1       // notice the min and max values have changed 0 and 1.
0   0   0   1   0   1
0   1   1   0   1   1
0   0   1   0   1   1
0   0   1   1   1   1
```

d) logFile (use argv[6]): To log the progress of your program.  Look at logFile can help you debug your program.

******************************

## III. Data structure:

******************************

- image class
    - (int) numRows
    - (int) numCols
    - (int) minVal
    - (int) maxVal
    - (int*) histAry //an 1D integer array, size of maxVal + 1; need to be dynamically allocated at run time.
    - (int**) imgAry // a 2D array, size of numRows by numCols; need to be dynamically allocated at run time.
    - (int) thrVal // use atoi (argv[2]) to convert string to integer; with #include <string> at the program header.

    Methods:

    - loadImage (imgAry, inFile) // load the content of input file to imgAry. **On your own**.
    - computeHist(...) // **See algorithm below**.
    - printHist (…) // Output histAry to histCountFile using the format given in the above. **On your own**.
    - dispHist (…) // Output histAry to histGraphFile using the format given in the above. **On your own.**
    - binaryThreshold (…) // **See algorithm below**.
    - PrettyPrint (…) // **See algorithm below**. This method will be used in many of the later projects.

******************************

## IV. main (...)

******************************

```
Step 0: check argc count is correct
        inFile ← open input file use argv[1]
        histCountFile, histGraphFile, binThrFile, logFile ← open from argv[3], argv[4], argv[5], argv[6]
        check all files can be opened.
Step 1:  numRows, numCols, minVal, maxVal ← read from inFile
Step 2:  imgAry ← dynamically allocate, size of numRows by numCols
         histAry ← dynamically allocate, size of maxVal+1, and initialize to 0, a must!
Step 3:  loadImage (imgAry, inFile)
         PrettyPrint (imgAry, logFile)
Step 4: ComputeHist (imgAry, histAry, logFile)
Step 5: printHist (histAry, histCountFile, logFile)
Step 6: dispHist (histAry, histGraphFile, logFile)
Step 7: thrVal ← get from argv [2]  // use atoi (…) function
Step 8: logFile ← "The threshold value =  " thrVal // write the value
Step 9: binaryThreshold (imgAry, binThrFile, thrVal, logFile)
Step 10:  close all files
```

***********************************************

## V. ComputeHist (imgAry, histAry, logFile)

***********************************************

```
Step 0: logFile ← “Entering computeHist ()
Step 1: i ← 0
Step 2: j ← 0
Step 3: val ← imgAry[i][j]
        if val < minVal or val > maxVal
            logFile ← “imgAry [i, j] value is not within minVal and maxVal” // write i and j so you know which pixel.
            exit (1)
Step 4:  histAry [val] ++
Step 5: j++
Step 6: repeat Step 3 to Step 5 while j < numCols
Step 7: i++
Step 8: repeat Step 2 to Step 7 while i < numRows
Step 9: logFile ← “leaving computeHist ()
```

*************************************************

## VI. binaryThreshold (imgAry, binThrFile, thrVal, logFile)

*************************************************

```
Step 0: logFile ← “Entering binaryThreshold ()”
        logFile ← “The result of the binary thresholding using ” thrVal “ as threshold value” // write thrVal
        binThrFile ← numRows, numCols, 0, 1   // new image header
        logFile ← numRows, numCols, 0, 1   // new image header
Step 1: i ← 0
Step 2: j ← 0
Step 3: if imgAry [i, j] >= thrVal
                binThrFile ← write 1 follows by a blank
                logFile ← write 1 follows by a blank
         else
                binThrFile ← write 0 follows by a blank
                logFile ← write 0 follows by a blank
Step 4: j++
Step 5: repeat Step 3 to Step 4 while j < numCols
Step 6: binThrFile  ← new line
        logFile ← new line
Step 7: i++
Step 8: repeat Step 2 to Step 7 while i < numRows
Step 9: logFile ← “leaving computeHist ()
```

******************************

## VII. PrettyPrint (imgAry, logFile)

******************************

```
Step 0: logFile ← “Enter PrettyPrint ()”
Step 1: logFile ← output numRows, numCols, minVal, maxVal
Step 2: str ← convert maxVal to string // use to_string (…) function
        Width ← length of str
Step 3: i ← 0
Step 4: j ← 0
Step 5: logFile ← imgAry[i][j]
Step 6: str ← convert imgAry[i][j] to string
        WW ← length of str
Step 7:  logFile ← write one blank space
        WW ++
Step 8: repeat step 7 while WW <= Width
Step 9: j++
Step 10: repeat Step 5 to Step 9 while j < numCols
Step 11: i++
Step 12: repeat Step 4 to Step 11 while I < numRows
Step 13: logFile ← “leaving PrettyPrint ()”
```