

Project 1: Given a grey-scale image, you are to perform the following tasks:

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.

Language: C++

Project points: 10 pts

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

-0 2/14/2021 Sunday before midnight

-1 for 1 day late: 2/15/2021 Monday before midnight

-2 for 2 days late: 2/16/2021 Tuesday before midnight

-10/10: 2/16/2021 Tuesday after midnight

*** Name your soft copy and hard copy files using the naming convention as given in the project submission requirement discussed in a lecture and is posted in Google Classroom.

*** 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.

1. Run your program on data1 with threshold 5
2. Run your program on data2 with threshold 38.
3. Include in your hard copy *.pdf file as follows:

- Cover page.
- source code.
- 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.
- Output outFile3 for data 2.
- Output outFile4 for data 2.

I. Input: There are two inputs to the program.

a) inFile1 (`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]`)

II. Outputs: There are four output files.

- a) OutFile1 (use argv[3]): For the output of histogram in the following format (to be used in the future project):
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 hist(i)

For example:

```
4 6 1 12
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) OutFile2 (use argv[4]): Display the histogram (for visual) as follows:
first text line is the image header then follows by a list of : greyScale (numpixels): number of +'s
for example, the output of the histogram of the above image would be:

Use the maximum of 70 +'s for all counts greater than 70. Use small font size so that 70 +'s can be printed on one text line.

```
4 6 1 12
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) outFile3 (use argv[5]): The result of the threshold of the input image. (To be used for future processing.)

Note: The output binary image also needs to have the image header.
For example, given the above image and 6 as the threshold value then the binary image would be:

```
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) outFile4 (use argv[6]): (For nice visual purposes).

For example, given the above threshold image, the pretty print replace 0 with a period.

```
4 6 0 1
. . . 1 . 1
. 1 1 . 1 1
. . 1 . 1 1
. . 1 1 1 1
```

III. Data structure:

- image class

```
- numRows (int)
- numCols (int)
- minVal (int)
- maxVal (int)
- histAry(int*) //a 1D integer array, size of maxVal + 1
                  // need to be dynamically allocated at run time

- thresholdValue (int) // via argv[2]
```

Methods:

```
- computeHist(...) // The algorithm is given in the lecture note
- printHist (...)// on your own; see the above example
- dispHist (...)// on your own; see the above example
- threshold(...) // The algorithm is given below
```

IV. **main** (...)

step 0: inFile ← open input file use argv[1]
open all 4 outFile via argv[3], argv[4], argv[5], argv[6]

step 1: numRows, numCols, minVal, maxVal ← read from inFile

step 2: histAry ← dynamically allocate and initialize to 0

step 3: ComputeHist (...)

step 4: printHist(outFile1)

Step 5: dispHist (outFile2)

step 6: close inFile
reopen inFile

Step 7: thrVal ← get from argv[2]
outFile3 ← "The threshold value uses is " thrVal
outFile4 ← "The threshold value uses is " thrVal

Step 8: threshold (inFile, outFile3, outFile4, thrVal)

step 9: close all files

```

*****
V. threshold (inFile, outFile3, outFile4, thrVal)
*****
Step 0: minVal  $\leftarrow$  0
        maxVal  $\leftarrow$  1

Step 1: outFile3, outFile4  $\leftarrow$  output numRows, numCols, minVal and maxVal

Step 2: pixelVal  $\leftarrow$  read from inFile one integer at a time

Step 3: if pixelVal  $\geq$  thrVal
        outFile3  $\leftarrow$  write 1 follows by a blank
        outFile4  $\leftarrow$  write 1 follows by a blank
    else
        outFile3  $\leftarrow$  write 0 follows by a blank
        outFile4  $\leftarrow$  write . follows by a blank

Step 4: repeat step 2 to 3 until the inFile is empty

```