Project 6.1 (in C++): You are to implement two of the four run-length encoding methods taught in class. \*\*\* project 6.2 will include the other two methods.

Method 1) with zero and no wrap-around.

Method 4) No zero and wrap-around.

What you need to do for this project:

1) Your program will open the input image file and do all the initialization.

2) The encode method will be given in argv[2]

3) Your program call the given method to perform compression on the image file and outputs the result to a text file.

4) The name of the compressed file is to be created during the run-time of your program, using the original input file name with an extension “ \_EncodeMethodN,” where N is the method number in which the user entered. For example, if the name of the original image is “image”, and N is 1 (use method 1) then the name of the compressed file should be “image\_EncodeMethod1”. (This can be done simply using string concatenation.) \*\*\* -6 pts for hard code file name in the program!!!

5) To begin, you implement method 1 from the algorithm steps given in class, then run your program with image1. Check the output to see if it encoded correctly. If it does, then implement method 4.

6) When both methods work correctly, run your program 2 times, first method 1 and next method 2.

7) Then, run your program with image2 as you did for image1.

8) Include in your hard copies (as pdf files):

(a) Input image file

(b) image1\_EncodeMethod1

(c) image1\_EncodeMethod4

(d) image2\_EncodeMethod1

(e) image2\_EncodeMethod4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. Language: C++

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Points: 12 pts

Due Date: Soft copy and pdf hard copies: 3/22/2020 Sunday before midnight

1 day late: -1 pt 3/23/2020 Monday before midnight

2 days late: -3 pts 3/24/2020 Tuesday before midnight

-12 pts: after 3/24/2020 Tuesday after midnight

\*\*\* Name your pdf file using the same format as your soft copy, but

using extension: \_HardCopy; for example if your name is Joe Golden

your soft copy would be GoldenJ\_Project6\_CPP

your pdf hard copy would be GoldenJ\_Project6\_HardCopy

\*\*\* All on-line submission MUST include Soft copy and pdf hard copy in the same email with file name convention as given in the above.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. a) inFile (argv[1]): a txt file representing an image, where

the first text line has 4 integers, representing the image header:

numRows numCols minVal maxVal follows by rows and cols of pixel values separated by space.

Example-1 for gray-scale image,

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

Example-2 for binary image,

4 6 0 1 // image has 4 rows,6 cols, min is 0, max is 1

0 1 1 0 0 1

1 1 1 1 1 1

0 1 1 1 0 0

0 0 0 0 0 0

b) whichMethod (argv[2]) // let the program knows which method to be used.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

II. Out files: All compressed file names are generated during the run-time

The format of a compressed image as follows.

20 15 0 9 // header information (20 rows, 15 cols, min is 0, max is 9)

2 // method 2 was used

1 4 8 10 // startRow is 1, startCol is 4, color is 8, 10 pixels long

2 4 7 5 // startRow is 2, startCol is 4, color is 7, 5 pixels long

:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. Data structure:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- runLength class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- whichMethod (int)

- nameEncodeFile (string)

- encodeMethod1 (...) // The algorithm was given in class.

- encodeMethod4 (...) // Modify encodeMethod2, you should know how.

- skipZero (...) // skip zero to read the next pixel

- define other methods or variables as needed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. main (…)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: inFile 🡨 open argv[1]

outFile1 🡨 open argv[3]

step 1: numRows, numCols, minVal, maxVal 🡨 Read from inFile

step 2: whichMethod 🡨 from argv[2]

step 3: nameEncodeFile 🡨 argv[1] + “\_EncodeMethod” + “whichMethod”

step 4: encodeFile 🡨open (nameEncodeFile)

step 5: output numRows, numCols, minVal, maxVal to encodeFile

output whichMethod to encodeFile

step 6: case of whichMethod

case 1: encodeMethod1 (inFile, encodeFile)

case 4: encodeMethod4 (inFile, encodeFile)

default: error message

Step 7: close all files

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V. encodeMethod4 (inFile, encodeFile) // No zero & Wrapped around

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The algorithm steps may contain bugs, you are responsible to debug it.

step 1: row 🡨 0

Step 2: col 🡨 -1

length🡨 0

step 3: lastVal 🡨 skipZero (inFile, row, col, zeroCnt)

step 4: output row, col, lastVal to encodeFile

step 5: length++

Step 6: nextVal 🡨 skipZero (inFile, row, col, zeroCnt)

step 7: if zeroCnt > 0

output row, col, lastVal to encodeFile

lastVal 🡨 0

Step 8: if nextVal == lastVal

length ++

else

output length to encodeFile

lastVal 🡨 nextVal

length 🡨 0

step 9: repeat step 4 to step 8 until eof (inFile)

Step 10: output length

step 11: closed all files

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(int) skipZeros (inFile, row, col, zeroCnt)

// row, col, zeroCnt are reference variables, i.e., when any change here, are also changed in the calling method)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The algorithm steps may contain bugs, you are responsible to debug it.

step 0: zeroCnt 🡨 0

step 1: col++

pixelVal 🡨 read the next pixel from inFile \*one integer at a time\*

step 2: if pixelVal

zeroCnt ++

step 3: if col >= numCols

col 🡨 -1

row ++

step 4: repeat step 1 – step 3 until currVal != 0

step 5: return pixelVal // non-zero