Project 6: You are to implement all of the four run-length encoding and decoding methods taught in class in this project:

Method 1) with zero and not wrap-around.

Method 2) Not zero and not wrap-around.

Method 3) with zero and wrap-around.

Method 4) Not zero and wrap-around.

What you need to do for project 6 (runLengthEncodeDecode):

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

2) The program asks the user if he/she want to use another method to compress the file:

(‘Y’ for yes, ‘N’ for no.)

if the user entered ‘N’, exit the program

if the user entered ‘Y’, you ask which method to be used.

if method is not within 1 – 4

ask the user enters again

3) Your program call 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 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 3 then the name of the compressed file should be “image\_EncodeMethod3”. (This can be done simply using string concatenation.)

5) Close the compressed file.

// At this point, your should see two files in your directory: image, and image\_EncodeMethod3.

6) To make sure your encoding method works correctly, your program will open the compressed file (after it is closed) and your program will call runLengthDecode method3 to perform the de-compression. Your program outputs the de-compressed result to a text file. (If your program works correctly, this file should be the same as the original input image file.)

7) The name of the de-compressed file is created during run-time, using the name of encoded file with an extension “\_Decoded”. For example, if the name of the encode file name is image\_EncodeMethod3, then the name of the de-compressed file should be “image\_EncodeMethod3\_Decoded”.

8) Closed the de-compressed file.

// after this step your directory should have three files: image, image\_EncodeMethod3, and image\_EncodeMethod3\_Decoded.

9) Repeat 2) to 8) until user type “N” to exit the program. // you should enter, 1, 2, 3 and 4 for encoding.

10) You will be provided with one image file to test your runLengthEncodeDecode program.

11) Include in your hard copies:

(a) Print the input image file

(b) Print the result files of the four methods of encoding and decoding:

image\_EncodeMethod1, and image\_EncodeMethod1\_Decoded

image\_EncodeMethod2, and image\_EncodeMethod2\_Decoded

image\_EncodeMethod3, and image\_rEncodeMethod3\_Decoded

image\_EncodeMethod4, and image\_EncodeMethod4\_Decoded.

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

Language: C++

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

Soft copy due date: 3/26/2019 Tuesday before Midnight

Early submission +1 deadline: 3/24/2019 Sunday before Midnight

-1 pt due: 3/27/2019 Wednesday before midnight

After 3/27/2019, -12 pts for all students who did not submit soft copy

Make sure you submit your soft copy to your TA and cc to Dr. Phillips

Hard copy due date: 3/28/2019 Thursday in class,

-1 pt for late hard copy submission on 3/29/2019 Friday (under door A218).

All projects without hard copy after 3/29/2019 will receive 0 pts even you have submit soft copy on time and even if it works.

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

I. Input (argv[1]): a txt file representing an image (gray-scale or binary), where

the first text line has 4 integers, representing the "header" of the input image

numRows, numCols, minVal, maxVal, follows by rows and cols of pixel values (integers).

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

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

II. The format for the result of an encoded image is given below.

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

:

--------

1) outFile1 (argv[2]): to store the method used to compress the input, # of runs use in encoding and total bytes used.

For example:

Method 2 was used to decompress the input image

18 runs produced in Run-Length encoding

288 bytes used without counting the image header

2) outFile2 (NOT from argv): is created during the run-time, to store EncodeMethod1.

3) outFile3 (NOT from argv): is created during the run-time to store EncodeMethod1\_Decoded.

4) outFile4 (NOT from argv): is created during the run-time, to store EncodeMethod2.

5) outFile5 (NOT from argv): is created during the run-time to store EncodeMethos2\_Decoded.

6) outFile6 (NOT from argv): is created during the run-time, to store EncodeMethod3.

7) outFile7 (NOT from argv): is created during the run-time to store EncodeMethod3\_Decoded.

8) outFile8 (NOT from argv): The name is created during the run-time, to store EncodeMethod4.

9) outFile9 (NOT from argv): is created during the run-time to store EncodeMethod4\_Decoded.

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

III. Data structure:

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

- runLength class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- whichMethod (int)

- nameEncodeFile (string)

- nameDecodeFile (string)

- yesNo (char)

- runLengthEncode (…) // algorithm given below

- encodeMethod1 (…) // you should know how to code this method.

- encodeMethod2 (…) // you should know how to code this method.

- encodeMethod3 (…) // you should know how to code this method.

- encodeMethod3 (…) // you should know how to code this method.

- runLengthDecode (…) // algorithm given below

- decodeMethod1 (…)// you should know how to code this method.

- decodeMethod2 (…)// you should know how to code this method.

- decodeMethod3 (…)// you should know how to code this method.

- decodeMethod4 (…)// you should know how to code this method.

- define other methods or variables as needed.

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

IV. main (…)

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

step 0: inFile 🡨 open argv[1], the input image files

outFile1 🡨 open argv[2]

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

step 2: Display on the console of those four methods of choice (use the format given in the above.)

step 3: yesNo 🡨 ask user if he/she want to use another method to encode a file

if yesNo == ‘N’

exit the program

else

whichMethod <-- ask the user’s choice from the console

if whichMethod is not within 1 – 4

ask the user enters again

step 4: nameEncodeFile 🡨 argv[1] + “\_EncodeMethod” + “whichMethod”

encodeFile 🡨open (nameEncodeFile)

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

output whichMethod to encodeFile // on next text line

step 6: runLengthEncode (inFile, encodeFile, whichMethod) // see algorithm steps below

Step 7: closed encodeFile

re-open encodeFile

step 8: nameDecodeFile 🡨 nameEncodeFile + “\_Decoded”

decodeFile 🡨open (nameDecodeFile)

step 9: runLengthDecode (encodeFile, decodeFile) // see algorithm steps below

step 10: close encodeFile and decodeFile

step 11: repeat step 2 to step 12 until user enter “N” and program exit

step 12: close all files

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

V. runLengthEncode (inFile, encodeFile, whichMethod)

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

step 1: case of whichMethod

case 1: encodeMethod1 (inFile, encodeFile)

case 2: encodeMethod2(inFile, encodeFile)

case 3: encodeMethod3 (inFile, encodeFile)

case 4: encodeMethod4 (inFile, encodeFile)

step 2: Write one text line, include: whichMethod used, numRuns and total bytes, to outFile1.

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

VI. runLengthDecode (encodeFile, decodeFile) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

step 2: write numRows, numCols, minVal, and maxVal to decodeFile

step 3: whichMethod 🡨 read from encodeFile

step 4: case of whichMethod

case 1: decodeMethod1 (inFile, encodeFile)

case 2: decodeMethod2(inFile, encodeFile)

case 3: decodeMethod3 (inFile, encodeFile)

case 4: decodeMethod4 (inFile, encodeFile)