Project 6.2 (C++): You are to implement two of the four run-length \*decoding methods in this project:

Method 1) with zero and not wrap-around.

Method 4) Not zero and wrap-around.

// Remark:

// \*\*\* I wrote the algorithm steps for deCodeMethod1 in the specs for you,

// \*\*\* but you need to write the algorithm steps for deCodeMethod4 yourself;

//\*\*\* you must include your deCodeMethod4 algorithm steps (after main algorithm steps) in your cover sheet!

// \*\*\* The implementation of this project MUST follows algorithm steps given in the specs,

//\*\*\* as well as, the algorithm steps you wrote for deCodeMethod4. -5 pts for not doing so.

// \*\* You are NOT allow to a 2D array to store data!!!!

What you need to do for this project:

- First, as you did for your project 6.1,

- Run your 6.1 project on image1 using method1

- Run your 6.1 project on image1 using method4

- Run your 6.1 project on image2 using method1

- Run your 6.1 project on image2 using method4

// So that your director will have 4 encoded files, i.e.,

- image1\_EncodeMethod1

- image1\_EncodeMethod4

- image2\_EncodeMethod1

- image2\_EncodeMethod4

- Next, you will run 6.2 program four times: for each of the encoded files listed above as follows:

1) Your program open an input encoded file (argv[1])

2) whichMethod is given on the 2nd text line in the encoded file.

3) Your program call deCodeMethodN to do de-compression and outputs the result to a text file. N is the method used, 1 or 4 for this project.

4) 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 image1\_EncodeMethod1, then the name of the de-compressed file should be “image1\_EncodeMethod1\_Decoded”. (This can be done simply using string concatenation.)

5) Closed the Decoded file. If your program work correctly, image1\_EncodeMethod1\_Decoded should be the same as image1

6) Do the same for the other three encoded files.

7) Include in your hard copies (in pdf file format with the same name convention as given as before):

- Cover page (including algorithm steps for deCodeMethod4)

- Print image1\_EncodeMethod1

- Print image1\_EncodeMethod1\_Decoded

- Print image1\_EncodeMethod4

- Print image1\_EncodeMethod4\_Decoded.

- Print image2\_EncodeMethod1

- Print image2\_EncodeMethod1\_Decoded

- Print image2\_EncodeMethod4

- Print image2\_EncodeMethod4\_Decoded.

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

Language: C++

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

Points: 10 pts

Due Date: Soft copy and pdf hard copies: 3/29/2020 Sunday before midnight

1 day late: -1 pt 3/30/2020 Monday before midnight

2 days late: -3 pts 3/31/2020 Tuesday before midnight

-10 pts: after 3/31/2020 Tuesday after midnight

\*\*\* Name your pdf file using the same naming convention as given prior.

\*\*\* All on-line submission MUST include Soft copy and pdf hard copy in the same email with proper file names.

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

I. encodeFile (argv[1]): an encoded text file. The format 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

:

II. decoded file: The name of the decoded file is generated during the run-time

-6 pts if you hard the file names.

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

III. Data structure:

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

- runLength class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- whichMethod (int)

- nameEncodeFile (string)

- nameDecodeFile (string)

- startRow (int)

- startCol (int)

- color (int)

- length (int)

// A run in the run-length encoding: tartRow, startCol, color, length

- leftEndPrint (int) // the left end of the printing on the same column, may be used in deCodeMethod4

- rightEndPrint (int) // the right end of the printing on the same column, may be used in deCodeMethod4

// \*\* You are NOT allow to declare a 2D array to store data!!!!

Methods:

- deCodeMethod1 (…) // Algorithm steps are given below.

- deCodeMethod4 (…)// write your own algorithm steps. See the remark below.

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

IV. main (…)

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

step 0: nameEncodeFile 🡨 argv[1]

encodeFile 🡨 open (nameEncodeFile)

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

step 2: whichMethod 🡨 read from encodeFile // on the 2nd text line

step 3: nameEncodeFile 🡨 nameEncodeFile + “\_Decoded”

decodeFile 🡨open (nameDecodeFile)

step 4: decodeFile 🡨 output numRows, numCols, minVal, maxVal to decodeFile

step 5: case of whichMethod

case 1: decodeMethod1 (encodeFile, decodeFile)

case 4: decodeMethod4 (encodeFile, decodeFile)

default: error message

exit program

step 6: close all files

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

V. decodeMethod1 (encodeFile, decodeFile)

// encoded with zero and not wrap-around.

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

// This algorithm may contain bugs, debugging is yours

Step 0: RowCnt 🡨 0

Step 1: colCnt 🡨 0

Step 2: startRow, startCol, color, length 🡨 read from encodeFile // get a run

Step 3: decodeFile 🡨 output length times of color to decodeFile

Step 4: colCnt ++

Step 5: if colCnt >= numCols

deCodeFile 🡨 print end of text line

colCnt 🡨 0

rowCnt ++

Step 6: repeat step 2 to step 5 while rowCnt < numRows and not EOF (encodeFile)

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

VI. decodeMethod4 (encodeFile, decodeFile)

// encoded without zero and wrap-around.

Step 0: RowCnt 🡨 0

Step 1: colCnt 🡨 0

newLength 🡨 0

Step 2: startRow, startCol, color, length 🡨 read from encodeFile // get a run

Step 3: if (startRow !=0 or startCol != 0)

Calculate

newLength = startRow \* 22 + startCol

decodeFile🡨output length time of “0” to decodeFile in the begining

Step 4: decodeFile 🡨 output length times of color to decodeFile

Step 5: startRow, startCol, color, length 🡨 read from encodeFile // get a run

Step 6: if (rowCnt == startRow and colCnt != startCol)

Calculate

newLength = startCol – colCnt

decodeFile🡨output length time of “0” to decodeFile

Step 6: if (rowCnt != startRow or colCnt != startCol)

Calculate

newLength = (startRow-rowCnt-1)\*22 + (22-colCnt) + startCol;

decodeFile🡨output length time of “0” to decodeFile

Step 7: repeat step 4- step 6 while rowCnt < numRows and not EOF (encodeFile)

Step 8: check if (rowCnt < numRows and colCnt != 21)

Calculate

newLength = (startRow-rowCnt-1)\*22 + (22-colCnt) + startCol;

decodeFile🡨output length time of “0” to decodeFile

VII addZero(length, decodeFile)

Step 0: decodeFile 🡨 output “0” to decodeFile

Step 1: colCnt++

Step 2: if colCnt >= numCols (or colCnt % numCols == 0)

deCodeFile 🡨 print end of text line. ( cout << endl)

colCnt 🡨 0

rowCnt ++

Step 3: repeat Step 0 – Step 2 until i < length;

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

- Remember to print zeros in the beginning if startRow and startCol is not 0 and 0

- Remember to print zeros in the beginning of a row if startCol is not 0

- Remember to print zeros between two runs, if there are zeros between two runs

- Remember to break wrap-around when it occurs and continue printing in the next row

- You may use leftEndPrint and rightEndPrint for easier coding.