CV Spring 2021 Exam 2 (Part - 2 online Programming)

*** Exam 2 Part-2 is a closed book exam. However, you may access to your projects 1, 2, 3, 4, 5, and 6 which are stored in your computer, but NOT someone's previous programs.
*** YOU MAY NOT use internet to search anything for answers for this exam, if you do so, your exam score for this part will be -999.
*** There are two online programming questions below: Q40 and Q41. Implement Q40 in C++ and Q41 in Java.
===

Q40 (C++): Run-length encoding is a method for image compression. There are four options in run-length encoding, depending on whether to include 0's or not; and whether to wrap around or not. You are going to write a program to do the easiest encoding option: including zero and NO wrapped around. (See input and output example below.) (40)

**** What you need to do for submission *****
a) Implement the encoding program.
b) Run your program once with Q40_data1 and once with Q40_data2 in the email you received from Dr. Phillips.
c) Name your soft copy: LastnameFirstInitial_Q40_cpp.zip
d) Name the hard copy: LastnameFirstInitial_Q40_PDF.pdf
** include in your hard copy:
- cover page (without algorithm steps)
- Q40 source code
- encodeFile for Q40_data1
- deBugFile for Q40_data1
- encodeFile for Q40_data2
- deBugFile for Q40_data2

e) Submit the soft copy and hard copy in the same email to your TA and cc to Dr. Phillips
with email subject: (CV) your first name your last name <Exam 2 part 2: Q40 submission>

************** Q40 Specs *******************

I. inFile (argv[1]): a txt file representing an image with image header.
For example:

```
5 9 0 9                    // image has 5 rows 9 columns, min is 1, max is 9
0 0 4 4 4 4 4 4 4
4 4 4 4 4 3 3 3 3
3 3 5 5 5 7 7 7 7
2 2 2 2 2 0 0 9 9
0 0 0 0 0 0 0 0 0
```
****
II. There are two output files:
a) encodeFile (argv[2]): the result of encoding, in text file format.
For example: Below is the result of encoding of the above inFile.
(Note: Your do not need to write the comments in your encodefile.)

```
5 9 0 9  // image header
0 0 0 2  // startRow is 0, startCol is 0, color is 0, 2 pixels long
0 2 4 7  // startRow is 0, startCol is 2, color is 4, 7 pixels long
1 0 4 5  // startRow is 1, startCol is 0, color is 4, 5 pixels long
1 5 3 4   // startRow is 1, startCol is 5, color is 3, 4 pixels long
2 0 3 2
2 2 5 3
2 5 7 4
:
```

b) deBugFile (argv[3]): to write debugging statements for partial credits if your program does not work completely.

****
III. Data structure:
****

```
- Encode class
        - (int) numRows
        - (int) numCols
        - (int) minVal
        - (int) maxVal
        - (int) startRow
        - (int) startCol
        - (int) greyScale
        - (int) length
        - (int **) img // 2D array to store input image, to be dynamically allocated at run time
                     // For easy implementation we use this array; in reality, 2D array is not needed in encoding.
        - loadImg (…) // re-use your previous code
        - encodeOneRow (row, encodeFile, deBugFile) // the method encodes one row of img at a time.  On your own.
                                 // You may request this method's algorithm steps from Dr. Phillips
                                 // for giving up 15 points of this question.
        - You may define other methods or variables as needed.
```

****
IV. main (…)
****

```
Step 0: inFile ← open argv [1]
        encodeFile ← open argv [2]
        deBugFile ← open argv [3]
Step 1: numRows, numCols, minVal, maxVal ← Read from inFile
        img ← allocate a 2D array
Step 2: encodeFile ← output numRows, numCols, minVal, maxVal to encodeFile
        deBugFile ← ← output numRows, numCols, minVal, maxVal to deBugFile // with caption
Step 3: loadImg (inFile)
Step 4: row ← 0
Step 5: encodeOneRow (row, encodeFile, deBugFile)
Step 6:: row++
Step 7: repeat step 5 to step 6 while row < numRows
Step 8: close all files.
```

==
Q41 (Java): Write a decoding program to transform a run-length encoded file to its original image. (25)

**** What you need to do for submission *****

a) Implement the decoding program, see the spec below.
b) Run your program twice: once with Q41_data1 and once with Q41_data2 in the email you received from Dr. Phillips.
c) Name your soft copy: LastnameFirstInitial_ Q41_JAVA.zip
d) Name the hard copy: LastnameFirstInitial_ Q41_PDF.pdf
** include in your hard copy:
- cover page (without algorithm steps)
- Q41 source code
- decodeFile for Q41_data1
- deBugFile for Q41_data1
- decodeFile for Q41_data2
- deBugFile for Q41_data2

e) Submit the soft copy and hard copy in the same email to your TA **and cc to Dr. Phillips**
with email subject: (CV) your first name your last name <Exam 2 part 2: Q41 submission>

DO NOT use qc email to submit Q41. Use one of your email that does not reject Java zip file.

************* Q41 Specs ******************

I. inFile (argv[1]): a run-length encoded file with image header.

For example:

```
5 9 0 9  // image header
0 0 0 2
0 2 4 7
1 0 4 5
1 5 3 4
2 0 3 2
2 2 5 3
2 5 7 4
:
```

****

II. decodeFile (argv[2]): the result of the decompressed image, in text file format.
For example: Below is the result of decoding of the above inFile.

```
5 9 0 9                    // image has 5 rows 9 columns, min is 1, max is 9
0 0 4 4 4 4 4 4 4
4 4 4 4 4 3 3 3 3
3 3 5 5 5 7 7 7 7
:
```

****

III. Data structure:

****

- deCode class
    - (int) numRows
    - (int) numCols
    - (int) minVal
    - (int) maxVal
    - (int) startRow
    - (int) startCol
    - (int) greyScale
    - (int) length
    - deCoding (...) // see algorithm below.

    - You may define other methods or variables as needed

****

IV. main (…)

****

```
Step 0: inFile ← open argv [1]
          decodeFile ← open argv [2]
         deBugFile ← open argv [3]
Step 1: numRows, numCols, minVal, maxVal ← Read from inFile
Step 2: decodeFile ← output numRows, numCols, minVal, maxVal to e3codeFile
          deBugFile ←output numRows, numCols, minVal, maxVal to deBugFile // with caption
step 3: deCoding (inFile, decodeFile) // see algorithm below
Step 4: close all files
```

****

V. deCoding (inFile, decodeFile, deBugFile)

****

```
Step 0: Row ← 0
          deBugFile ← output "entering deCoding method"
Step 1: colCnt← 0
Step 2: startRow, startCol, greyScale, length  ← read from encodeFile // get a run
          deBugFile ← startRow, startCol, greyScale, length, colCnt
Step 3: i ← 1
Step 4: decodeFile ← output greyScale follow by a blank to decodeFile
          deBugFile ← output greyScale follow by a blank to decodeFile
Step 5: i++
Step 6: deBugFile ← output i, row, colCnt, numCols and greyScale to deBugFile
Step 7: repeat Step 4 to 5 while i <= length
Step 8: colCnt += length
Step 9:  if colCnt >= numCols
                    decodeFile ← print end of text line
                    row++
Step 10: repeat step 1 to step 9 while row < numRows and not EOF (inFile)
Step 11: deBugFile ← output "exiting deCoding method"
```