Project 5 (in C++): You are to implement quadtree representation of a given image.

You will be given two images (img1 and img2) to test your program.
Run your program twice, one on each test image.
For each image, print the image on the top of a blank page, then draw the quadtree representation of the image on the bottom of the blank.

****
Language: C++
Project points:10 pts
Due Date: <u>Soft copy (*.zip) and hard copies (*.pdf)</u>:

+1 (11/10 pts): early submission, 10/23/2022, Sunday before midnight
-0 (10/10 pts): on time, 10/27/2022 Thursday before midnight
-1 (9/10 pts): 1 day late, 10/28/2022 Friday before midnight
-2 (8/10 pts): 2 days late, 10/29/2022 Saturday before midnight
(-10/10 pts): non submission, 10/29/2022 Saturday after midnight

Hard copy includes:

- Cover sheet (one page)
- The page has img1 on the top and its quadtree representation on the bottom (-1 pt if omitted.)
- The page has img2 on the top and its quadtree representation on the bottom (-1 pt if omitted.)
- Source code
- outFile1 for img1
- outFile2 for img1
- outFile1 for img2
- outFile2 for img2

****
I. Input: inFile(argv[1]) a text file contains a binary image with header information. The first text-line is the header information: numRows numCols minVal maxVal follows by rows and columns of pixels, 0's and 1's. An example below is a binary image has four (4) rows and five (5) columns, minVal is 0 and maxVal is 1:

```
4 5 0 1
0 0 0 1 0
0 1 1 0 1
0 0 1 0 1
1 0 1 1 1
```

****
II. Outputs:

a) outFile1 (arvs[2]): the pre-order
and post-order traversals of the quadtree
(needs caption for each traversal!)

b) outFile2 (argv[3]): all the debugging outputs to get some partial credits.

****

III. Data structure:

****

- QtNode class
    - (int) color // 0, 1 or 5
    - (int) upperR // the row coordinate of the upper left corner
      //of the image area in which this node is representing

    - (int) upperC // the column coordinate of the upper left corner
      //of the image area in which this node is representing
    - (int) Size
    - (QtNode*) NWkid // initialize to null
    - (QtNode*) NEkid // initialize to null
    - (QtNode*) SWkid // initialize to null
    - (QtNode*) SEkid // initialize to null

    Method:

    - constructor: (upperR, upperC, Size, NWkid, NEkid, SWkid, SEkid)
      // to create a QtNode

    - printQtNode (...) // output the given node's: color, upperR, upperC, NWkid's color, NEkid's color,
      //SWkid's color, SEkid's color), in one text line
- QuadTree class
    - (QtNode*) QtRoot
    - (int) numRows
    - (int) numCols
    - (int) minVal
    - (int) maxVal
    - (int) power2Size
    - (int **) imgAry
      // a 2D array, need to dynamically allocate at run time of size power2Size by power2Size.

    methods:

    - constructor (…) // performs necessary allocations and initializations
    - (int) computePower2 (...) // Algorithm is given below
      // The method determines the smallest box of power of 2 by power of 2
      // that fits the input image. It returns the
      // the size (one side) of the power2Size
    - loadImage (...) //load the input data onto imgAry, begins at (0, 0). On your own.
    - zero2DAry (...) // set the given 2D array to zero. Note: C++ does not initialize. On your own.
    - (QtNode*) buildQuadTree (...) // Algorithm is given below.
    - (int) addKidsColor (node) // add up node's 4 kids colors. On your own.
    - (bool) isLeaf (node) // returns true if node is a quadtree leaf node (color is 1 or 0),
      // otherwise returns false.
    - preOrder (...) // see algorithm steps below.
    - postOrder (...) // on your own.

```
******************************
IV. Main (...)
******************************
step 0: inFile, outFile1, outFile2 ← open from argv []

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

step 2: power2Size ← computePower2(numRows, numCols)

        output power2Size to outFile2 with caption

step 3: imgAry ← dynamically allocate the array size of power2Size by power2Size

step 4: zero2DAry (imgAry)

step 5: loadImage (inFile, imgAry)

        output imgAry to outFile2 with caption

step 6: QtRoot ← BuildQuadTree (imgAry, 0, 0, power2Size, outFile2)

step 7: preOrder (QtRoot, outFile1)

step 8: postOrder (QtRoot, outFile1)

step 9: close all files
```

```
******************************
V. (QtNode*) buildQuadTree (imgAry, upR, upC, size, outFile2)
******************************
Step 1: newQtNode ← get a new QtNode with:
                        (-1, upR, upC, size, null, null, null, null)
        printQtNode (newQtNode, outFile2) // for debugging, with caption

Step 2: if size == 1 // one pixel
                newQtNode's color ← imgAry[upR, upC] // either 1 or 0
       else
                halfSize ← size / 2
                newQtNode's NWkid ← buildQuadTree (imgAry, upR, upC, halfSize)
                newQtNode's NEkid ← buildQuadTree (imgAry, upR, upC + halfSize, halfSize)
                newQtNode's SWkid ← buildQuadTree (imgAry, upR + halfSize, upC, halfSize)
                newQtNode's SEkid ← buildQuadTree (imgAry, upR + halfSize, upC + halfSize, halfSize)

                sumColor ← addKidsColor (newQtNode)

                if sumColor == 0  // all 4 kids are 0
                        newQtNode's color ← 0
                        set all newQtNode's four kids to null
                        // newQtNode is now a leaf node

                else if sumColor == 4  // all 4 kids are 1
                                newQtNode's color ← 1
                                set all newQtNode's four kids to null
                                // newQtNode is now a leaf node
                else
                        newQtNode's color ← 5   // newQtNode is an internal node, grey.

step 3: return newQtNode
```

```
*******************************
VI. (int) computePower2 (numRows, numCols)
*******************************
step 0: size ← max (numRows, numCols)

step 1: power2 ← 2

step 2: if size > power2

                    power2 *= 2

step 3: repeat step 2 until size <= power2

step 4: return power2
```

```
*******************************
VI, void preOrder (Qt, outFile1)
*******************************
If  isleaf  (Qt)
   printQtNode (Qt)
else
       printQtNode (Qt)
       preOrder (Qt’s NWkid)
       preOrder (Qt’s NEkid)
       preOrder (Qt’s SWkid)
       preOrder (Qt’s SEkid)
```

```
*******************************
 VII. void postOrder (Qt, outFile1)
*******************************
 On your own
```