QuadTree project: The quadtree representation of a given binary image.

\*\*\*\* Language: \*\* C++ \*\*

Due Date: C++: Soft copy 3/18/2016 (Friday) before midnight

C++: Hard copy 3/21/2016 (Tuesday) before exam II

In addition to the hard copy of your program, you need to do the following:

Step 1: Print out the image and get a blank piece of paper

Step 2: Draw the root of the quardtree on the blank paper

Step 3: If the image is grey (color == 2)

3.1: divide/draw the image into four quads

and draw the four kids (NW, SW, SE, NE) under the node on

the quadtree paper

3.2: for each of the four kids, repeat step 3

until no more kid's color is 2.

step 4: Attach the printed image you draw

and the quadtree you draw) inside of the cover page.

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

I. Input: a text file contains a binary image (0/1) with image header information)

// Use argv[1] for the name of input file

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

II. Output1: a text file contains the pre-order of the quadtree from your program.

\*\*\*\* The name of the output file should be given in argv[2]

Output2: all the debugging outputs within the program should be written in output2.

\*\*\*\* The name of the output file should be given in argv[3]

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

III. Data structure:

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

- A QuadTreeNode class

friend QuadTree class

friend image class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- color (int) // 0, 1 or 2

- rowOffset (int)

- colOffset (int)

- NW\_kid (QuadTreeNode\*) // initialized to NULL

- SW\_kid (QuadTreeNode\*) // initialized to NULL

- SE\_kid (QuadTreeNode\*) // initialized to NULL

- NE\_kid (QuadTreeNode\*) // initialized to NULL

- methods:

- computeOffsets

- computenumRowsCols

- computeMinMaxValue // optional

- computeColor

- printNode

- constructor(s)

- destructor(s)

- etc.

- A QuadTree class

- rootNode (QuadTreeNode\*)

- methods:

- constructor(s)

- BuildQuadTree

- PrintQuadTree

- An image class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- imgAry (int \*\*) a 2D array, need to dynamically allocate at run time

of size numRows by numCols.

- methods:

- makeImgAry // dynamically allocate 2D imgAry

- loadImage //read the input data onto imgAry

- constructor(s)

// assign values to numRows,..., etc.

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

III. Algorithms

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

step 0: create program skeleton

//write up all the classes for the program.

step 1: open the input file

and read the image header (numRows, numCols, minVal and maxVal)

step 2: makeImgAry

step 3: loadImage

step 4: rootNode <-- Create the quadtree root

step 5: BuildQuadTree (in recursion, see algorithm steps below)

step 6: PrintQuadTree (in pre-order recursion, see algorithm steps below)

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

Recursive Algorithm for BuildQuadTree

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

void BuildQuadTree (QuadTreeNode \*T)

Step 1: if T is null or T's color != 2

return

else

create a QuadTreeNode for T->NW;

BuildQuadTree(T->NW)

create a QuadTreeNode for T->SW;

BuildQuadTree(T->SW)

create a QuadTreeNode for T->SE;

BuildQuadTree(T->SE)

create a QuadTreeNode for T->NE;

BuildQuadTree(T->NE)

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

Recursive Algorithm for PrintQuadTree

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

void printQuadTree (QuadTreeNode \*T)

step 1: if T is null

return

else

printNode(T) // see algorithm below.

printQuadTree (T->NW)

printQuadTree (T->SW)

printQuadTree (T->SE)

printQuadTree (T->NE)

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

void printNode (QuadTreeNode \*T)

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

print T's attributes in \*one text line\*

Example: if T's

color (2) numRows (32) numCols(32) minVal(0) maxVal(1) rowOffset(0) colOffset(32)

NW\_kid's color(1) SW\_kid's color(0) SE\_kid's color(0) NE\_kid's color(2)

Your output for the node T, in one text line, will be as below:

2 32 32 0 1 0 32 1 0 0 2