Assignment 2: Image Compression and Manipulation

Due: Monday January 29th, 23:59. Worth 3% of your course grade.

Web stuff: See the course web site for starter code, and a summary of any hints and announcements about the assignment.

Purpose: Practise writing recursive code (and learn some cool stuff about QuadTrees).

QuadTrees: A quadtree is a hierarchical data structure that can be used for representing images. It is based on the idea of recursively decomposing a two-dimensional space.

The root of a quadtree represents the entire two-dimensional image. If the entire image is black, the root has colour black and has no children (because the space needn’t be broken down further). Or if the entire image is white, the root has colour white and has no children. But if the image has a mixture of white and black, the root is assigned the colour grey, and has four children, each representing a quadrant of the whole image. Those children are defined in the same fashion.

In other words, the image is divided into quadrants and sub-quadrants until every block is made up entirely of black or white elements. Below is an example of an image and its corresponding quadtree.

In this case, I have ordered the quadrants (and hence the children of every node) this way: NW, NE, SE, SW. The letters “g”, “b”, and “w” in each node indicate colour grey, black, and white, respectively.

QuadTrees offer a space-efficient representation for images that contain large homogeneous areas. And certain operations on images can be done very efficiently using this representation.
Your Task: For this assignment, you will implement a QuadTree class with functions for the following operations:

- Read from the standard input a two-dimensional image and build a QuadTree that represents it.
- Print to the standard output the two-dimensional image represented by a QuadTree.
- Check whether a given location within the image represented by a QuadTree is black or white.
- Compute the area of the image represented by a QuadTree.
- Reverse the image represented by a QuadTree, turning black to white and vice versa.
- Rotate the image represented by a QuadTree.

For bonus marks, you may choose to implement the following additional operations:

- Lay one image over another, essentially computing their union.
- Find the intersection of two images.

The full specifications for each function are defined in prototypes provided in the starter code. The input and output format is defined below.

Starter code etc: You will be given the following on the course web page:

- QuadTree.h: Defines the prototypes for the required and bonus functions.
- A sample input or output file.

All the details about the algorithms, and about how to represent a QuadTree are up to you; I have only constrained you in providing the prototypes.

If you do not implement the bonus functions, just define them with an empty body.

You may add members to QuadTree.h, but you must not change the prototypes I provide. If you do, your program may fail all of the autotesting.

I/O Format: All images either read by your program or printed by your program will be of size $2^r$ by $2^r$, where $r$ is an integer $\geq 0$. The format for such images must be as follows:

$2^r$ lines, each consisting of $2^r$ characters and ending in a newline. Each of the characters must be either ‘0’ (to indicate white) or ‘1’ (to indicate black).

See the web page for a sample file.

How to submit: Details on how to submit your assignment will be available on the web site. This time, you will be asked to submit a make file, so I suggest that you start using one from the beginning.