Assignment #3 Report

Description of Quadtree Implementation

Because I used the C programming language for this assignment, I chose to use C's structs to represent my quadtrees. The tree itself consisted of structs called nodes. Each node contained a pointer to the "currentcell", or the partition that this node described, and an array of pointers to the children of this node. Each "currentcell" is represented as a cell struct, which in itself contains a "room" pointer (called "myroom") and four integer variables. The integer variables represent the indices of the edges of this partition, and they are named "leftwall", "rightwall", "topwall", and "bottomwall". The "room" pointer inside each "cell" struct contains two "point" pointers named "topleft" and "bottomright". These points are "point" structs, which contain two integer variables ("x" and "y") each.

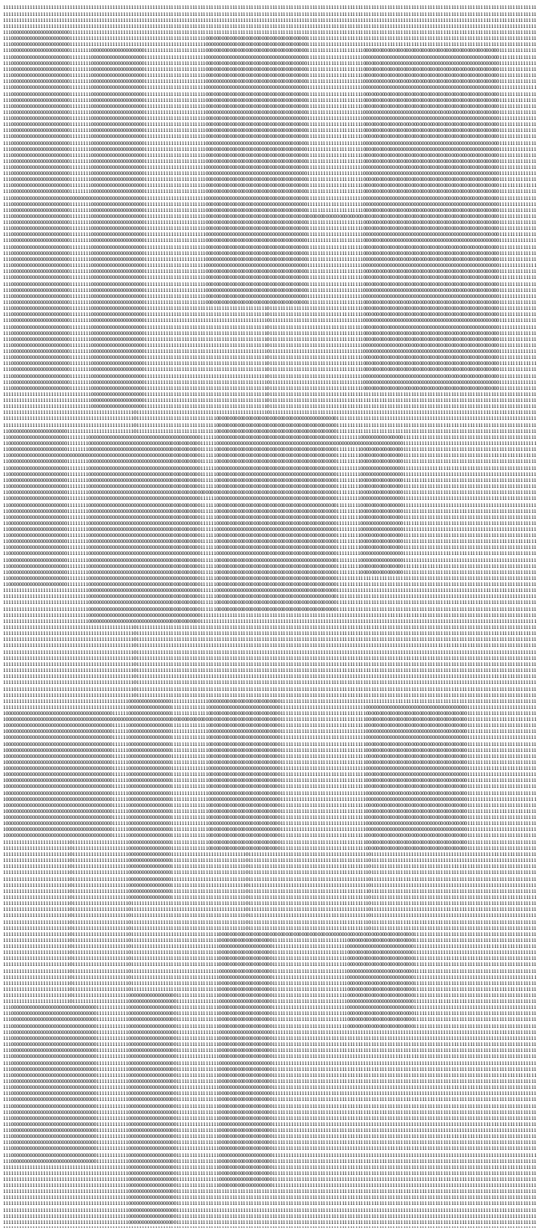
The only major design decision that I made was the decision to represent the data in a way that is easy to understand at the cost of some time and memory efficiency. I made this decision because I knew that optimizing the space requirement and time requirement of running the program was not necessary due to the fact that this program is not expected to ever be used for a large-scale project.

The conversion from quadtree to binary matrix begins at the very beginning of the program; a 2D integer array is created and filled with 1's at the beginning of the program. The array is partitioned (represented by the integer variables in the "cell" struct) until the partitions are all below a certain size, then rooms are created by filling choosing a random x-coordinate and y-coordinate for the "topleft" and "bottomright" points of leaf nodes only. Constraints are placed on what qualifies as a valid partition or coordinate to avoid rooms being too small, and any offenders are rerolled until they qualify as valid. Once all rooms are created, the rooms are "drawn" onto the binary matrix; that is, a function changes all 1's between the "topleft" and "bottomright" coordinates to 0's in the binary matrix. Once the rooms are drawn, a random number determines how they will be connected; with four rooms arranged in a rectangular pattern, it takes at least three hallways to guarantee that all rooms are reachable. The random value determines which of the four possible hallways is not drawn, then a different function connects the rooms via drawing a hallway to connect them in a random location that has the minimum distance to connect.

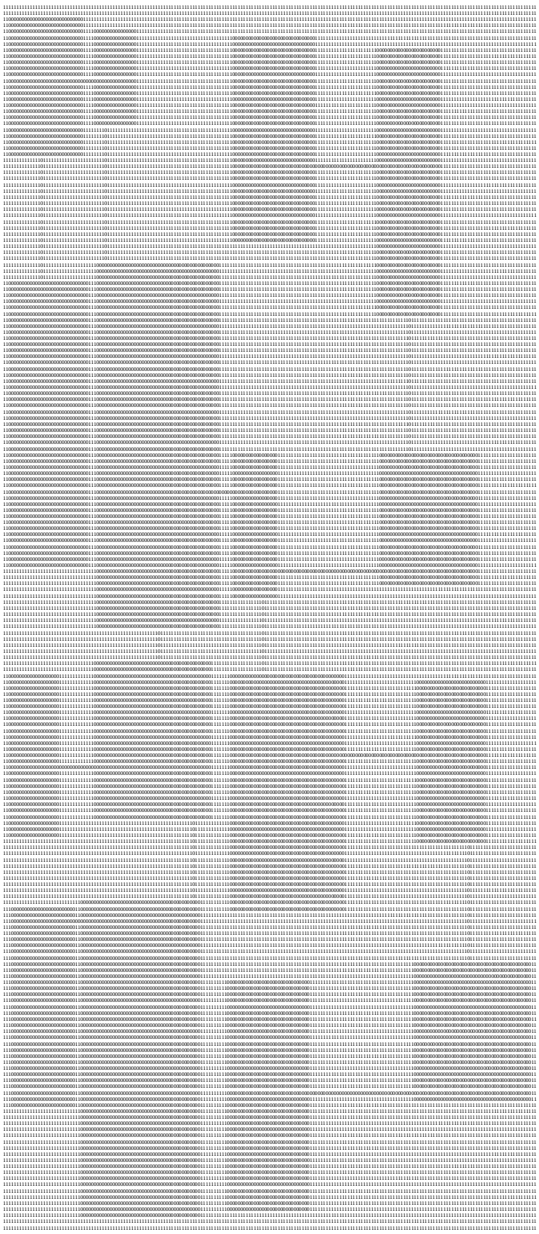
In case you want to see what the tree generated actually looks like, I created a "printTree" function. I commented it out for the purposes of this assignment, but you could always uncomment that line in the main function to see what the generated quadtree actually looks like (represented in text).

Five Generated Maps

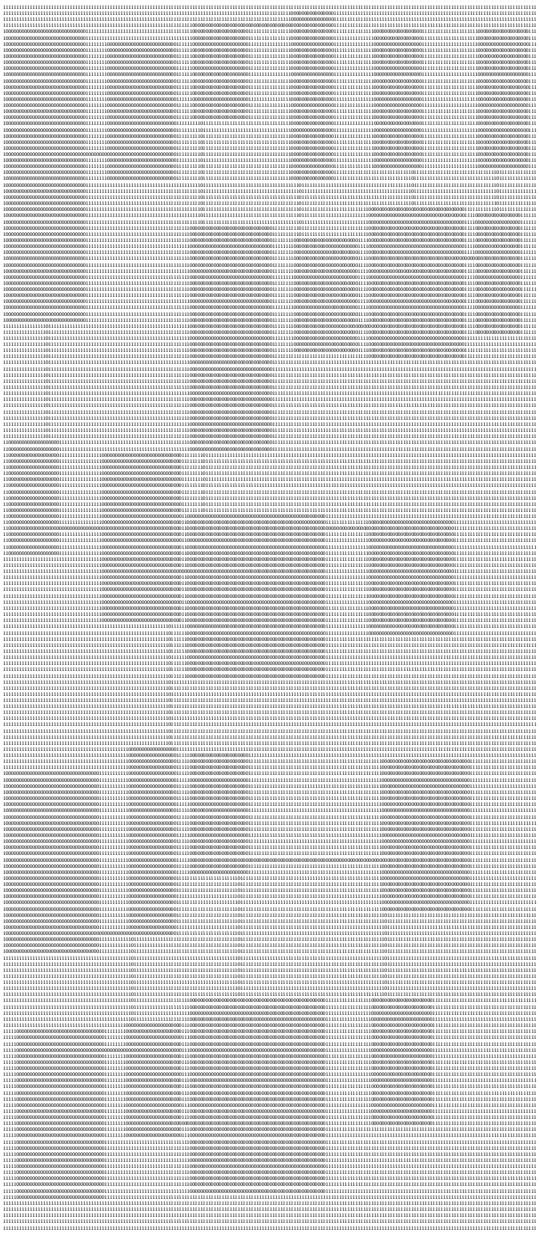
Map 1



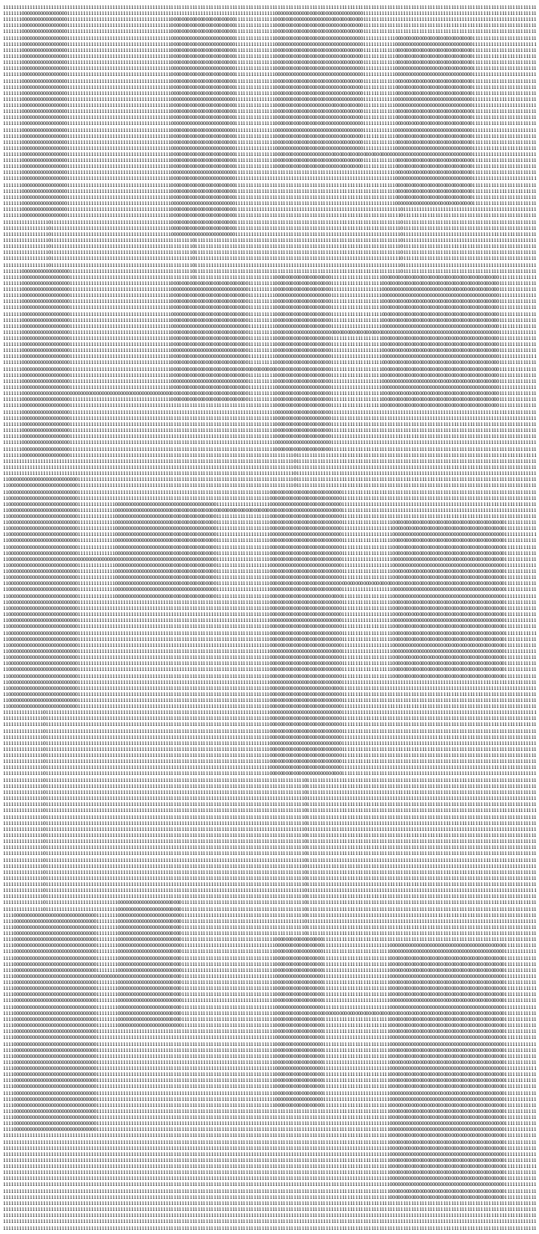
Map 2



Map 3



Map 4



Map 5

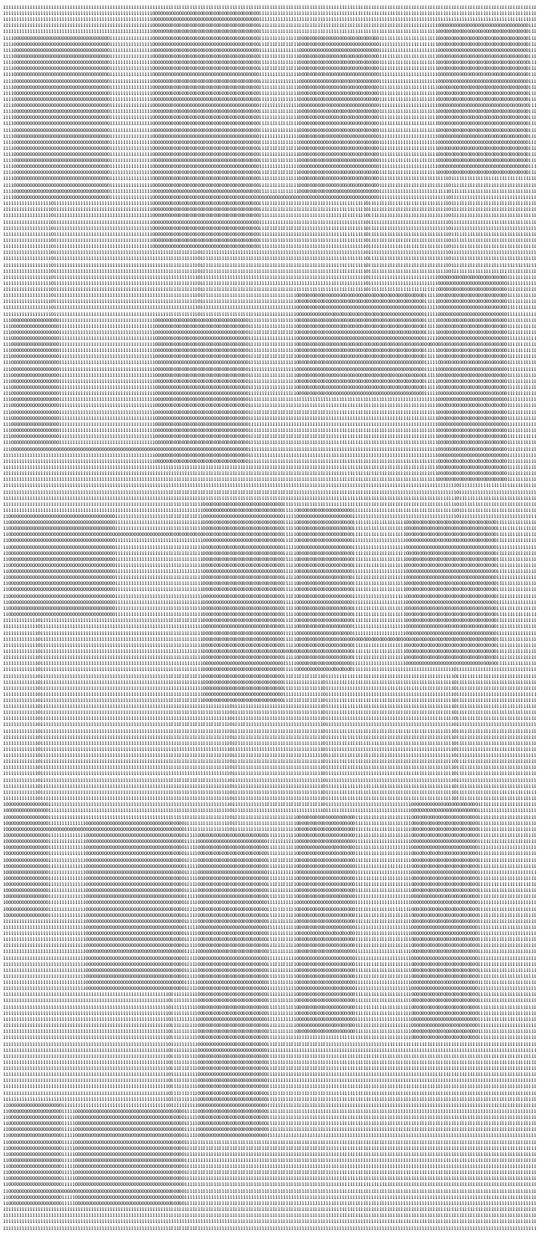
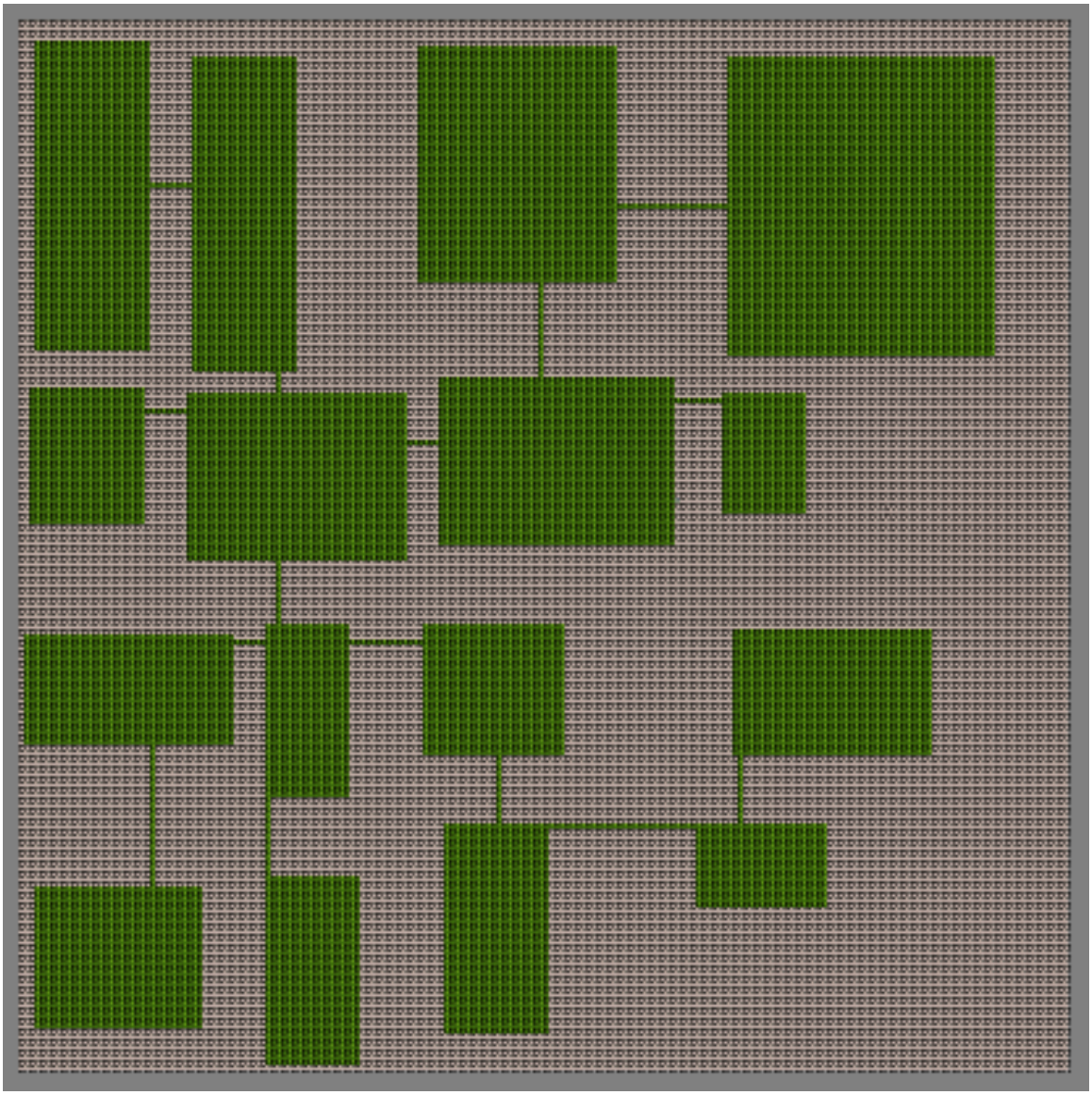


Image of Themed Map 1



While the dimensions may have not held very well in the text form, the above image is an exact replica of *Map 1* created in *Tiled*.

Conclusion

Looking back, I believe that this assignment was the perfect level of challenging and fascinating. I enjoyed doing this assignment greatly, and the freedom to complete it in C (my "native" programming language) was relieving. I look forward to using the skills gained from this assignment in my future work.