Evan Arends

Design Doc Extra Credit

I designed this extra credit using pattern matching and guards. Before I started coding, I had to extensively look through the notes on heap to make sure I knew what I was coding. I also almost did not realize I was coding a min heap instead of a max heap so its good I double checked. These programs was built extensively using recursion, which Haskell excels at as well as with applied sub programs. When I say applied sub programs I mean I needed to make sub programs besides the ones required to assist me with the main programs. Haskell makes it so it is hard to write one long chunk of code and instead you must break it up into many sub functions. For example, I needed to make a function that checked if the heap was full as well as if the root node had both kids. The challenges to this program were learning how to combine the functions together. This proved a small hiccup of understanding layering and tracing what the program was supposed to do before coding the actual thing. The next hard part of this program was dealing with the many cases that a heap can represent for example if it’s a full heap, if it only has one child, and the layers beyond that. Haskell also having your nodes written out in mostly word form in a line is hard to visualize, so I was constantly having to draw the heap representation on paper to make sure the program was working correctly. For testing my program, I had to test not just small heaps but big heaps as well with heights of 3-5 to make sure the trickle down and replacement functions were working correctly. Testing takes a bit of time to do because you have to build up the heap and you have to write out on paper what the heap is supposed to look like to make sure the program works correctly. Further testing included a large variety of numbers to make sure all possibilities of code were being executed. I wanted to make sure ‘insert’ and ‘deleteMin’ worked for all lengths of nodes, and node counts. So often times I would build up a tree one by one and every time I added a node, I would call the function on the tree to make sure it worked for a tree with 0 nodes all the way up to a tree with 15-20 nodes. The possibilities are endless so I tried to test as many combiantions, height, and node counts as possible. Overall Haskell has moved up my list on favorite languages past c++ and right behind python at number 1.