Edward Cruz – COSC 3333 – HW4: Programming Assignment 2

# Experiment 1

**Display the root and the nodes/letters in its right subtree in the BST on one line as a set of letters and not a tree; display the nodes in the first half of the array storing the Heap on the next line.**

**What pattern do you see?**

The BST displays the same if not more letters than the first half of the heap on a more than likely average.

**Explain your take on this pattern and how you can or cannot justify it as a reliable pattern.**

In most cases I've seen that the BST displays the same number of letters if the root happens to be at the later end of the alphabet. If the root happens to be about the first 60-70% of the alphabet, it is practically guaranteed to be bigger than half the heap. I cannot justify this pattern being reliable as I have already run into a test case that breaks my observation's pattern with the small pool of 11 words I used for test purposes.

# Experiment 2

**Is this Heap identical to the Heap that you created straight from the user input?**

No, it seems that the levels are preserved but the order within the level and sibling are at risk of being flipped. I did not see a scenario where the entire tree is flipped or anything of that nature. It seems to be an issue only pertaining at a sibling scope. If the siblings are flipped then of course this can cause the rest of the tree to be flipped/mirrored from that level onward. From my observations, however the Tree display of the Heap was always the same shape and only had flipped siblings.

**Does it matter how the Heap looks like?**

I would say yes because if the Heap formed from the BST Array did not come out at least with the same tree structure display as the original then that would be a major indicator of not all nodes being transferred across the manipulations in creating the Heap. The reason as to why is because the heap uses a specific procedure to display the nodes from the array to a tree like structure. If they were not the same, then it would mean that the Heap had either been manipulated with insertions/removes before its construction or the coping of the array over to the heap was majorly flawed and had skipped over some nodes. (Note: I realized after the fact that these statements are only true because of the way I inserted the Letters in the original Heap. The original was a gapless array and organized using trickleUp() at each insertion.)

**Discuss your take on whether you find it necessary for the Heap to be configured a certain way for it to function properly and whether their configuration of nodes can impact their algorithmic applications.**

The book mentions how to sort a heap by using an array to dump the heap into an array with remove() so that trickleDown() gets applied to every subtree as nodes leave. Then insert it back into the heap and removing them again should return a sorted array. If trickleDown and trickleUp are configured properly then it shouldn't matter for the heap. It can be fixed by just repeating the steps above so that it can be ready for algorithmic applications.