COSC 222 Lab 5 – Heap Sort

In this lab you will implement a heap structure by completing a class called `MinHeap` which extends a `FullBinaryTree` and also simultaneously implements the `HeapADT` interface. Further, the `MinHeap` class uses a generic type E which, itself, extends `Comparable` (meaning that `MinHeap` can be instantiated with any generic type E as long as E is a `Comparable` type, like a number or String or anything having a `.compareTo()` method on it).

The important functionality you will have to implement are the methods `upHeap` and `trickleDown`.

upHeap: This is called when a new element is added to the heap. A new element is first added at the end of the tree, and then `upHeap` will compare it to is parent node to see if it should come before the parent and swap upward. If it does swap upward, `upHeap` should be called again to further check if it needs to move higher.

`trickleDown`: After extracting the minimum element (the root node), the largest (last) element should replace the root node (thus reducing the final index by 1), and then this large value at the root should trickle down the tree to an appropriate position. In order to properly trickle down, it should check if it has a `leftChild` and a `rightChild` and the smaller of the two children (if they exist) should be swapped with the current position, if they come before it. When the current position swaps down to its child position, it should trickle down again from there.

The wiki article contains an example of inserting an element in a binary heap and removing an element from a binary heap:

https://en.wikipedia.org/wiki/Binary_heap

The `UseExample.java` gives an example of how to use this `MinHeap` to perform HeapSort. Test your code by sorting a variety of arrays, and variety of types (ints, doubles, Strings, … anything comparable).