**Part 3: Binary Heap**

In project 1, you have implemented a Leaderboard where N scores were processed one at a time and an array was used to maintain a set of the best M scores. You implemented this both with a sorted array and with an unsorted array. Now you will re-implement it using an array-based binary heap. You will need to implement *delMin* and *insert*, both of which should take O(logM) time in the worst case. Recall the property of Min-heap: A[Parent(i)] <= A[i]

For part 3, you will be implementing four methods. We have provided some helper functions to help you with your implementation but you are not required to use them. You need to implement the following methods:

* *Swim:* If the heap order is violated because a node's key becomes smaller than that node's parent's key, then swap the node with its parents. Keep swimming up the heap until reaching a node with a smaller key or the root.
* *Sink:* If the heap order is violated because a node's key becomes larger than one or both of its children's keys, then swap the node with the smaller one of its children. Keep sinking until reaching a node that has two larger children.
* *delMin:* Remove the minimum key from the heap and restore the min-heap condition.
* *Insert:* Insert the new key in the heap and restore heap condition.

(Hint 1: The easiest implementation index-wise is to ignore index 0. )

(Hint 2: The pseudocode from HW 1, Problem 5 may be useful!)

Run the test cases by right clicking on the MinHeapTest.java. If you are using Intellij, you can press Alt+Enter on the variables that are red. It helps you import packages to support the test frame. The test cases here only test for the overall output. If you are having problems with a specific function, you may write you own test cases or add print statements (System.out.println) or use the debug tool provided by you IDE.