AP Computer Science Name_____
Heap Sort

Description

Previous O(n log n) sorts were the MergeSort and the Quicksort. The Heapsort's advantage over MergeSort and Quicksort is that the best, average, and worst case run time are all O(n log n). This means that the order of the input elements does not significantly affect the run time, unlike MergeSort and Quicksort, which in the worst case can degrade to $O(n^2)$.

HeapSort is not good for small n, because of the overhead needed to rearrange elements to make a heap.

Heapsort requires no temporary storage space.

Algorithm

The elements are stored and coded in an array starting at index 1, but it's useful to think of them arranged in a tree. A heap with h elements has h/2 subtrees.

1. Transform the random array into a maxHeap. Starting from the last element and working upward, compare the child to its parent. If greater, swap them. (You can do this either iteratively or recursively.) The greatest element is at the root!

2. The greatest element is at the root! Swap the root and the last element. Then walk down the heap from the top, swapping with the larger child, until it is either in place or at the end. (You may do this either iteratively or recursively.) Reduce the last index by one, swap, and walk down again. After N-1 iterations, the array is ordered, least to greatest. Voila! On the next page is a heap. Perform a heapSort, least to greatest:

Look for the maximum value (it's at the root), swap it to the end, and re-form the heap.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	
	99	80	85	17	30	84	2	16	1	start
										swap first and last
										swap first and last
										swap first and last
										swap first and last
										swap first and last
										swap first and last
										swap first and last
										swap first and last -- Done

Ordering this sequence took 18 steps. By the O(n*log n) formula, $9 * \log_2 9 = 28.53$.

Assignment

Using the Sort lab from earlier in the year, add the Heapsort in to the mix. Display it. Make it into a heap. Display it, heap sort it, and display it again. You will need to add some methods like these:

```
swap(int[] array, int a, int b)
heapDown(int[] array, int k, int size)
sort(int[] array)
```