Heaps Lab #3

Complete and test the methods `insert` and `remove.` Verify that the given `heapSort` works Turn your code into Athena2 at the end of the class period, finished or not.

Add the following code to your `HeapUtils` class:

```
/**
 * heapSort is a O(nlogn) sort algorithm that begins with the data ordered as a heap.
 * The root node is swapped with the node n, and the new tree having one less element
 * is then heapified.  The process repeats until all items in the heap are processed.
 * @param heap is the array representation of a complete binary tree meeting the
 *        heap condition
 * @param heapSize is the size of the heap which may differ from the size of the array
 * precondition: heap is an array representation of a binary tree that satisfies the
 *               heap condition
 * postcondition: the array heap contains the data sorted from smallest to largest
 *                beginning at index 1.  The original heap is destroyed.
 */
  public static void heapSort(Comparable[] heap, int heapSize)
  {
     for(int i = 1; i <= heapSize; i++)
     {
        int index = heapSize +1 - i;  // last index is heapSize
        int length = heapSize - i;
        swap(heap, 1, index);
        heapify(heap, 1, length);
     }
   }
```

You may need to add a `swap` method, or hard-code the swap into the method.

Verify that `heapSort` works as intended.

Your grade will be based on what you turn in today.