HW 5: Heapsort

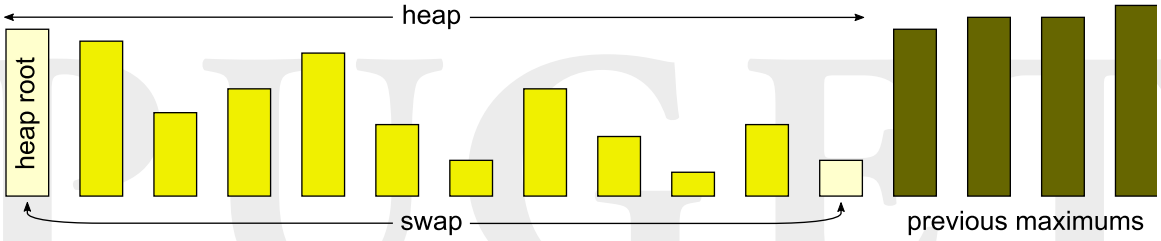
You must implement a heapsorting class, inheriting from the provided abstract `Sorter` class. Your class should be called `HeapSort`.

The `main()` method can be copied nearly verbatim from this week's lab. It should ask the user for a number, and then time how long (in ms) it takes to sort an array of that length:

```
Testing heapsort.
Please enter the array size: 50000
Array of size 50000 took 87 ms to sort.
```

As we have gone over in class, heapsort is a two-stage algorithm. It starts by making the array into a heap ("heapifying"), by first finding the last element with a child or children, and performing the "sink" operation on it. This will possibly swap the element with its largest child, and then its largest child, and so on, until the element is larger or equal to its children. It then iterates backward toward the root, each time doing a sink operation on element in question. After the root has undergone the sink, the array will be in a heap.

The second stage is to repeatedly swap the root element of the heap with the heap's last element. After this is done, the last element of the heap is no longer considered to be part of the heap. The new root is most likely too small, and so it must undergo a sink to bring it to its correct place. The process repeats until the array is sorted.



Both stages are easily shown to be linearithmic, or $O(n \log n)$ algorithms. It is slightly more difficult to prove that the heapify stage is actually linear. However, it is still dominated by the linearithmic second stage, and so the algorithm is itself linearithmic.

Please remember to catch all possible exceptions and to handle them intelligently, rather than allowing the user to see it!