3. **Minqueues.** Consider an extension of the queue abstract data type, called a minqueue, which supports operations ENQUEUE, DEQUEUE, and FINDMIN. Assume that the elements to be stored are integers (or any other totally ordered data type). FINDMIN simply returns the smallest element in the minqueue, but does not remove it. Using a standard implementation of a queue, the FINDMIN operation takes $O(n)$ time in the worst case. However, consider a more clever data structure for this abstract data type which works as follows: There are two regular queues used. The first queue is called the “real queue” and the second queue is called the “helper queue”. When the ENQUEUE($x$) operation is performed, the element $x$ is enqueued in the regular way on the real queue. The element $x$ is also enqueued at the end of the helper queue. However, if the element $y$ immediately “in front” of $x$ on the helper queue is larger than $x$, then $y$ is removed from the helper queue. This process of $x$ annihilating the element immediately in front of it is repeated until the element immediately in front of $x$ is less than or equal to it or $x$ is at the front of the helper queue.

(a) **Describe how the DEQUEUE and FINDMIN operation are implemented in this data structure.**

(b) **Give the worst case running time for each of ENQUEUE, DEQUEUE, and FINDMIN using this data structure.**

(c) **Give an argument using the potential method to show that any sequence of $n$ ENQUEUE, DEQUEUE, and FINDMIN operations requires only $O(n)$ time.**

(Problem taken from HMC CS181b, taught by Ran Libeskind-Hadas.)