Extra Credit #1: Amortization

Handed out on Feb. 14, due on Feb. 26 at the beginning of class. Remember: write your own answers and use English or pseudocode when algorithms are requested. Late homeworks will not be accepted (turn in whatever you have).

**Problem EC1.** Show how to solve Problem 1a on Homework 1 so that the *amortized* running time of `enqueue` and `dequeue` is $O(1)$ no matter what pattern of $n$ `enqueue`s and `dequeue`s is executed.

In other words, the user of your implementation may perform any series of $n$ `enqueue` and `dequeue` operations, and you must show how to solve Problem 1a so that over the course of these operations, your implementation only takes $O(n)$ time.

Show the implementation (which would also serve as an answer to Problem 1a) and argue why it has the above property.