Team SAK -- Jason Kim, Soojin Choi, Adrian Kloskowski
APCS2 pd8
HW#48 -- Heap o'Trouble
2018-05-14

Algorithm for add()

1) Assuming that we are "storing" this heap in an array, add the new element to the end of the array.
2) From there use the equation floor((newElementIndex - 1) / 2) to get the index of the parent.
3) From there, if the condition of the heap is met and is correct, then stop. If not, swap the two.
4) Repeat steps 2 and 3 until the condition is met or until the new element is at index 0.

Algorithm for remove()

1) First, we must check if the requested element to be removed is even in the heap. If it's not stop now.
2) Get the index of the to-be-removed element.
3) If that to-be-removed element has 2 children, compare the two and see which one can be promoted while keeping the condition of the heap. Don't consider the to-be-removed element in this comparison as it's going to be removed (as the name suggests). Swap the promoted element with that to-be-removed element.
4) If that to-be-removed element has 1 child, swap the two.
5) If that to-be-removed element has no children, just remove it from the array.
6) Repeat steps 2 to 5 until that element is successfully removed. (AKA: once step 5 actually activates)