David Sun
CMPS 102
Homework 2

Problem 1: Design 3 algorithms based on binary min heaps that find the kth smallest # out of a set of n #'s in time:

a) $O(n \log k)$

b) $O(n + k \log n)$

c) $O(n + k \log k)$

Note: For all problems, an array will be used to represent the heap.

a) For part a, we initially allocate an array of size k for our heap with all the elements initialized to $-\infty$ *(according to CLRS, a heap containing all duplicate keys can still be classified as a valid min-heap).* Then, we will make a pass through our array A from $A[1 \ldots length[A]]$. For each A_i where $i \in \{1 \ldots length[A]\}$, check if $-A_i$ is greater than the minimum of the heap. If such is the case, call remove on the heap, and insert $-A_i$ into the heap. We apply this operation for $length[A]$ iterations. Once we've scanned through all the elements in A, the kth smallest element should be $-1 * smallest(heap)$.
Correctness for a: If we want the kth smallest element, we would scan through the array and keep track of a k element list. If an element in the array A_i is less than the maximum element L_{max} in the list, replace L_{max} with A_i. Thus, by the end of one pass through the array, the k-element list will contain the k smallest elements in the array. However, the constraint in part a is that only the smallest element can be removed. In this case, we construct a heap using the negatives of the array elements. Thus if the *negative* of an array element is *greater than* the minimum heap value, then replace the minimum heap value with the negative of the array element. Since $\forall a, b \in \mathbb{R}$, if $a > b$ then $-b > -a$, replacing the minimum value in the heap containing negatives is equivalent of replacing the maximums in a max-heap. By the end of one pass through the array, the min-heap would contain the negatives of the k-smallest values with the k-th smallest of the array as the smallest value of the heap.