Lab 7

1. Starting with the values 1, 2, 4, 4, 5, 6, 9, 11, 12, 12, 17, do the following:
    a. Create a heap H in which these values are the keys.
    b. Perform the insertItem algorithm to insert the value 7 into H. Show all steps.
    c. Perform the removeMin algorithm on H and show all steps.
    d. Represent H in the form of an array A.
    e. Perform the array-based insertItem algorithm to insert 14 into A – show all steps.
    f. Perform the array-based removeMin algorithm on A – show all steps.

2. Carry out the array-based version of HeapSort on the input array

    [1, 4, 3, 9, 12, 2, 4]

    Show steps and outputs along the way. Make sure to distinguish between Phase I and Phase II of the algorithm.

3. Carry out the steps of the recursive algorithm BottomUpHeap for the input sequence 11, 5, 2, 3, 17, 24, 1

4. Draw an example of a MaxHeap whose keys are all the odd numbers lie in [1, 21] (with no repeats), such that the insertion of an item with key 14 would cause up-heap to proceed all the way up to a child of the root (replacing that child's key with 14).

5. Carry out the following variation on the HeapSort algorithm: For Phase I, instead of "heapifying" the input array, replace the input array with a new array, obtained by performing bottom-up heap (the iterative version). Then carry out Phase II in the usual way. Do this for the input array: [10, 3, 7, 6, 4, 12, 5, 2, 12, 10, 1].

6. (Optional) be a MinHeap storing n keys. Give an algorithm for reporting all the keys in T that are smaller than or equal to a given query key x (which is not necessarily in T). Note that the keys do not need to be reported in sorted order. You can assume that locating the parent and children of a node take O(1). Give an estimate of the running time of your algorithm.