Heaps, Heapsort - Practice

If in the exam you are asked to *delete* or *remove* or *extract* from a heap, it means delete the max element (if a max-heap) or the min element (in a min-heap). If asked to *delete x* (or *remove x* or *extract x*) then you must delete the element *x*.

Any question that would ask to modify/adapt an algorithm, would provide the original code/pseudocode for that algorithm.

You must be able to write the code for the methods discussed in class.

H1. Book (CLRS) – heaps, heapsort

    a. 6.1-1, 6.1-4, 6.1-5, 61.1-6 (CLRS 3-rd edition, page 153)
    b. From 6.2-1 to 6.2-6 (inclusive) (CLRS, page 156)
    c. 6.3-1, 6.3-2
    d. 6.4-1, 6.4-3
    e. 6.5-1, 6.5-2, 6.5-6, 6.5-7, 6.5-8, 6.5-9. Also questions 6.5-3 and 6.5-4 (the code for the methods referred in the questions, would be provided in the exam in this case).

H2. (Heap operations. No code needed.)
A1) (7 points) Insert **55** in the max-heap below.
Redraw the heap and leave white the node whose value did not change and show values only in the modified nodes.
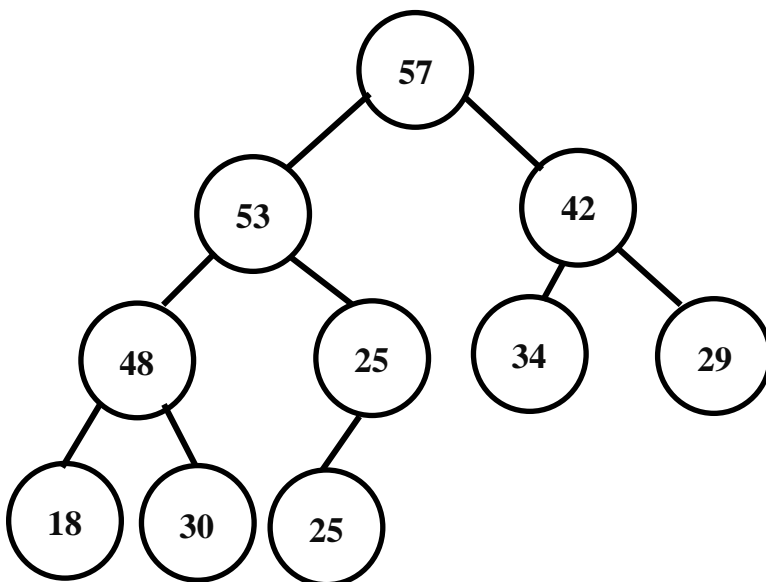
A2) Show the array that stores the above heap. Start at index 1.

A3) Rewrite the array (for the above heap) after each change it goes through during the `insert 55` operation.
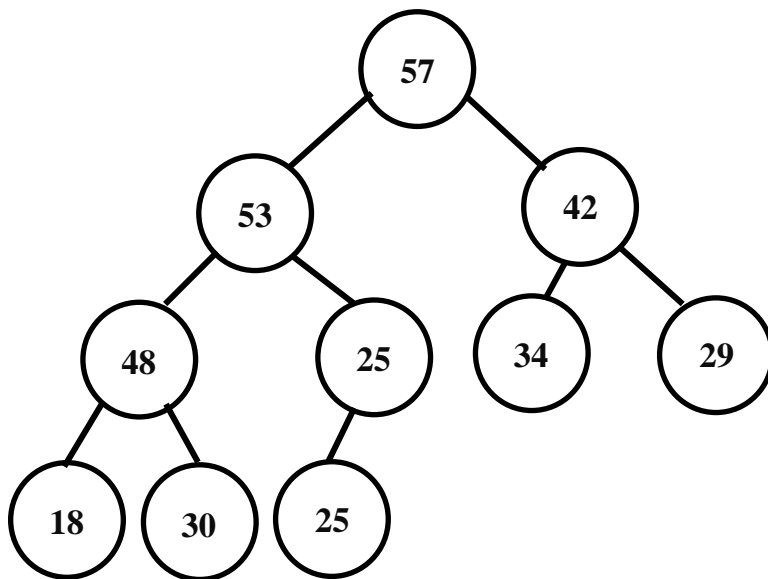
A4)

Version 1) Consider the heap AFTER 55 was inserted. Give the pairs (index, value) for every element of the heap that was modified (that is that is now different from what it was before the insertion). List the pairs in increasing order of the index. Separate them by comma. Do not put any empty spaces.

Version 2) Give the pairs of values that were swapped as part of performing the insert 55 operation. Separate the pairs by comma, do not put any empty spaces. List the pairs in the order in which the swaps were performed.

B1) (7 points) Perform one `remove()` operation on the heap below.
Redraw the heap and leave white the node whose value did not change and show values only in the modified nodes.

B2) Show the array that stores the above heap. Start at index 1.

B3) Rewrite the array (for the above heap) after each change it goes through during the `remove()` operation.

B4)

Version 1) Consider the heap AFTER the `remove()` operation. Give the pairs (index, value) for every element of the heap that was modified (that is that is now different from what it was before remove()). List the pairs in increasing order of the index. Separate them by comma. Do not put any empty spaces.

Version 2) Give the pairs of values that were swapped as part of performing the `remove()` operation. Separate the pairs by comma, do not put any empty spaces. List the pairs in the order in which the swaps were performed.

C1) Perform a `remove(48)` on the original heap from B1. Show the changes in drawing and as an array as done for B1, B2, B3.

Version 1) Give the (index,value) pairs to show the heap changes.

Version 2) Give the pairs of values that were swapped to show the heap changes:

H3. (14 pts) HEAPS

a) (8 pts) Does this array represent a Max-Heap with the root at index 1? (Ignore the data at index 0.)

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Data	12	78	50	30	8	65	22	28	4	6	12	2	13

1) (4 pts) Draw the heap like a tree.

2) (4 pts) Is this a valid heap?   Yes / No   **Briefly** justify your answer: