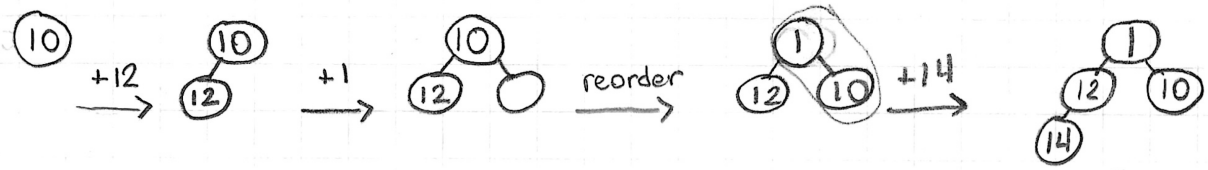
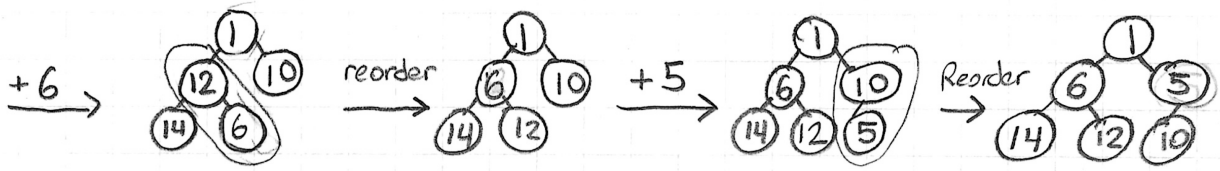
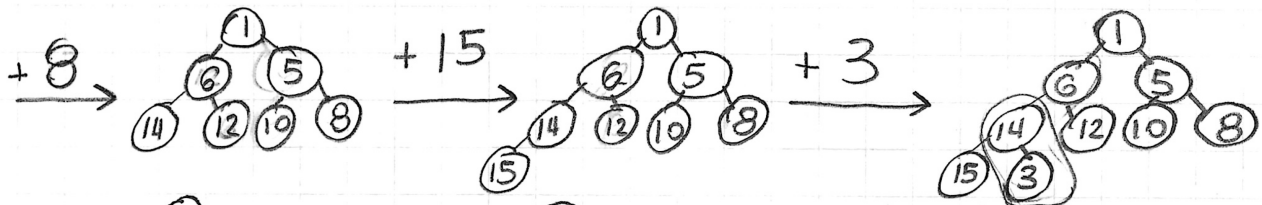
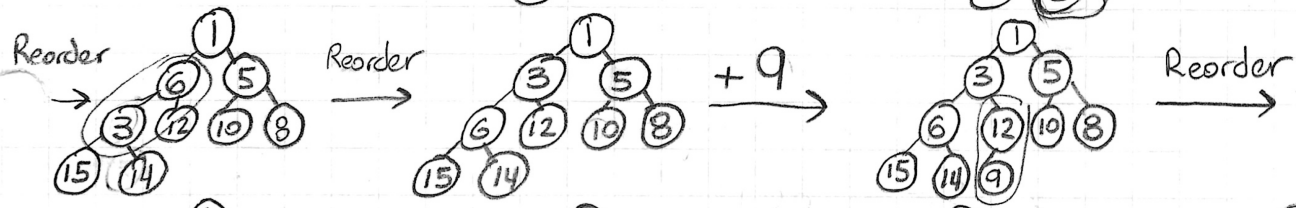
Problem 1)

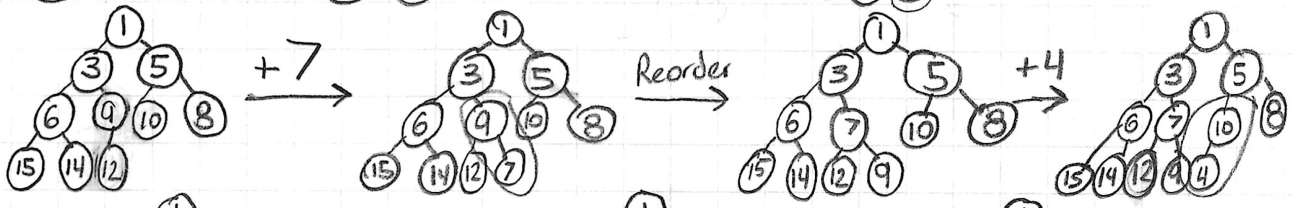
(a) 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, 2

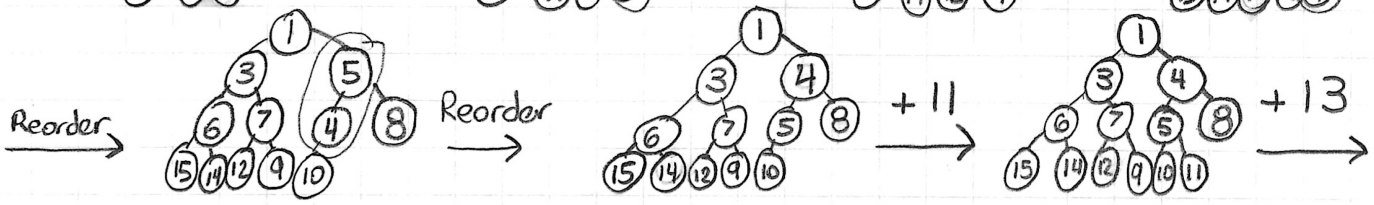


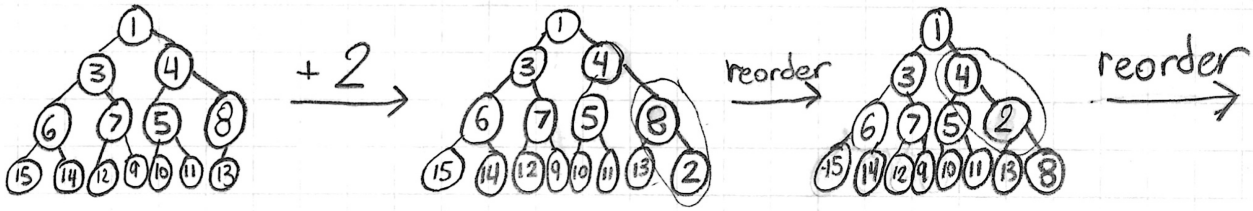


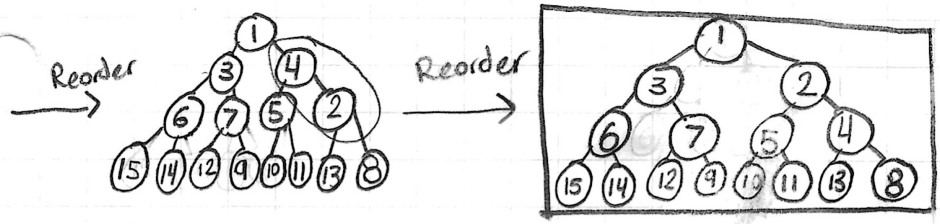




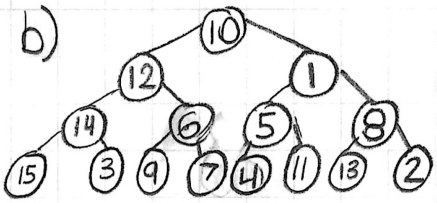




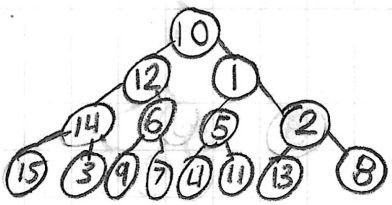




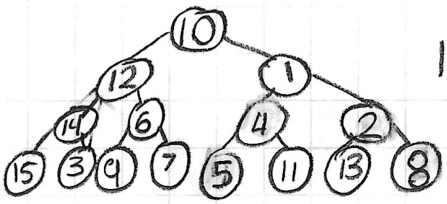
b)



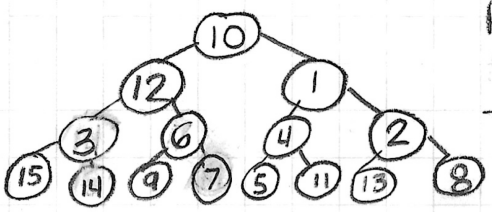
Percolate 8 Down



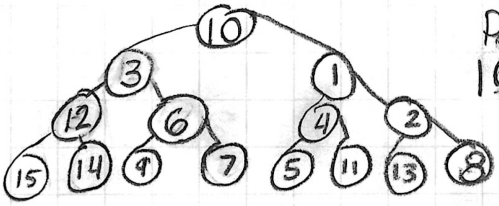
Percolate 5 Down



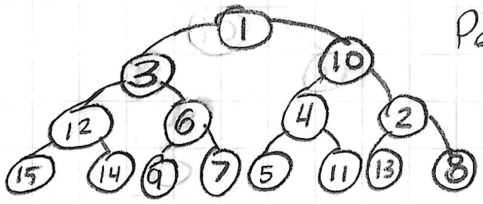
Percolate 14 Down



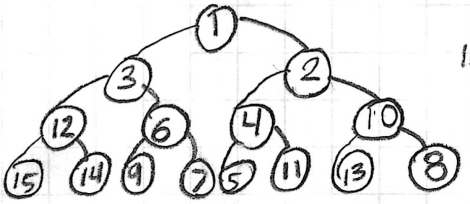
Percolate 12 Down



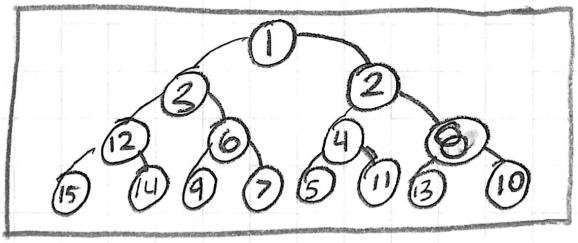
Percolate 10 Down



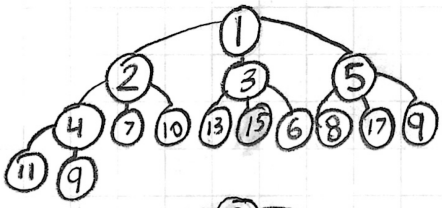
Percolate 10 Down



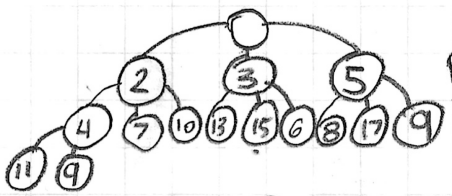
Percolate 10 Down



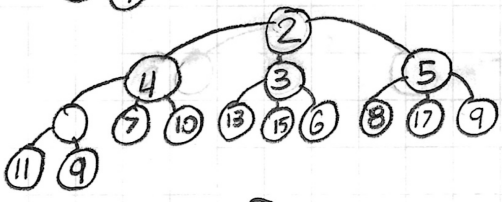
c)



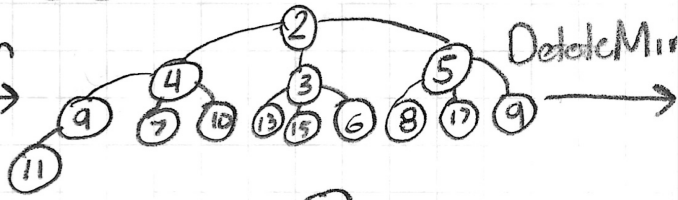
DeleteMin()



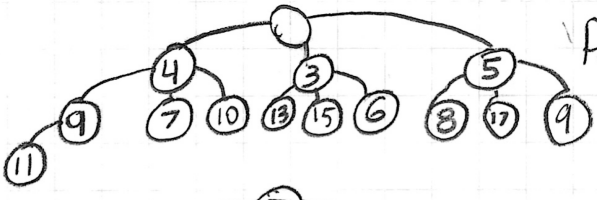
PercolateDown



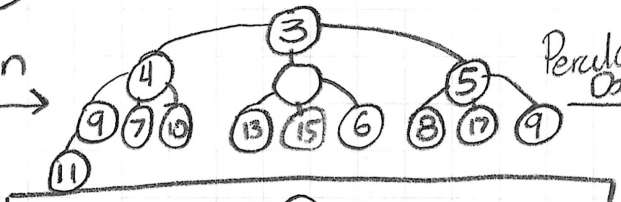
PercolateDown



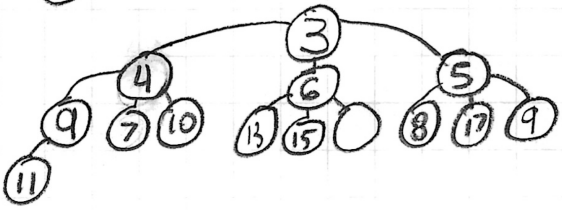
PercolateDown

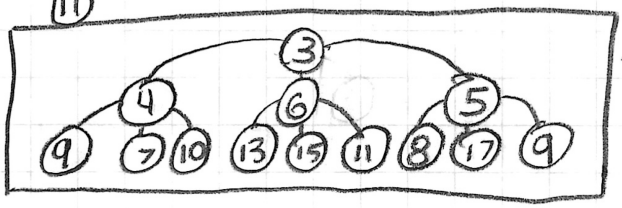


PercolateDown



Move 11



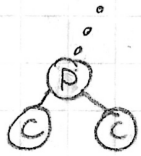


Problem 2)

(a) Show that the maximum item must be at one of the leaves.

↳ Property: A node must have its parent value smaller than itself, while its childrens value must be larger than itself.

∴ if a maximum value is not a leaf at the bottom, then the maximimvalue must have children which would be smaller, this is not possible due to the heap property of children always being bigger.



if p is maximum, c must be smaller since p is maximum. But, C cannot be smaller by the heap property.

(b) Show that any binary heap contains exactly $\lceil N/2 \rceil$ leaves.

↳ Property: each node in a heap will have one, two, or zero children.

I will prove by induction -> Base Case $N=1$ ← heap would only have 1 node.

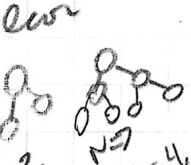
if $N=1$ then there are $N/2$ leaves in the heap in this case it is rounded to 1, therefore true.

Inductive Step $N = N+1$ therefore there will be $\frac{N+1}{2}$ leaves in a heap with $N=N+1$.

Assume N+1 is odd: removing 1 node should get rid of 1 leaf, at most. this was N even which add one making it $N+1/2$

Assume N+1 is even: removing 1 node should result in no exact change, ∴ it remains at N/2 which is the base case.

* in addition to this we know that the least leaf of a heap will always have space for N/2 nodes, therefore there must always be N/2 leaves

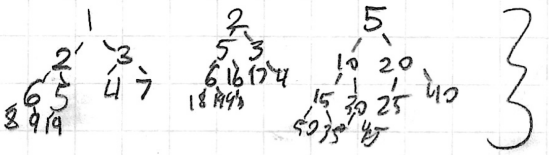


Problem 3)

(a) Done. 9, 8, 7, 6, 5, 4, 3, 2, 1, 19 → 1, 2, 3, 6, 5, 4, 7, 8, 9, 19

15, 16, 17, 18, 2, 3, 4, 5, 6, 98 → 2, 5, 3, 6, 16, 17, 4, 18, 15, 98

50, 45, 40, 35, 30, 25, 20, 15, 10, 5 → 5, 10, 20, 15, 30, 25, 40, 50, 45, 35



} the heaps of examples.

(b) The function will sort the array in decreasing order because it will rebuild it with the new root everytime, this in essence will isolate the current root, move it to the back and then rebuild and do it again

1, 2, 3, 4, 5 ... → ... 5, 4, 3, 2, 1

* Makes it a MAX heap

(c) The worst case running time will be $\Theta(\log N \cdot N)$

↳ Building the heap brings a $\Theta(N)$ running time into the picture.

↳ The loop goes 5, 4, 3, 2, 1 working with less entries everytime which is a characteristic of $\log(N)$.

$$\Theta(N) \cdot \Theta(\log(N)) = \Theta(N \cdot \log(N))$$

$$\sum_{j=1}^{size} \log(j) = \log((N-1)!)$$ } this is a popular sum theorem.