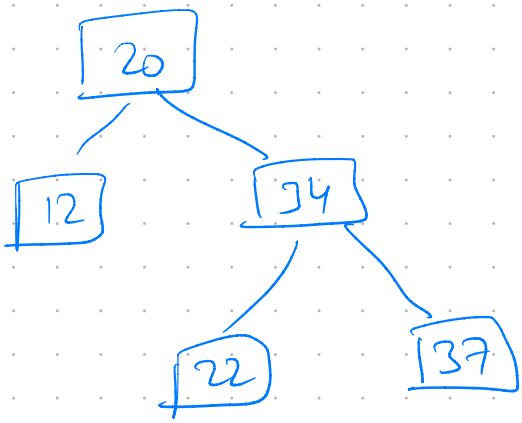
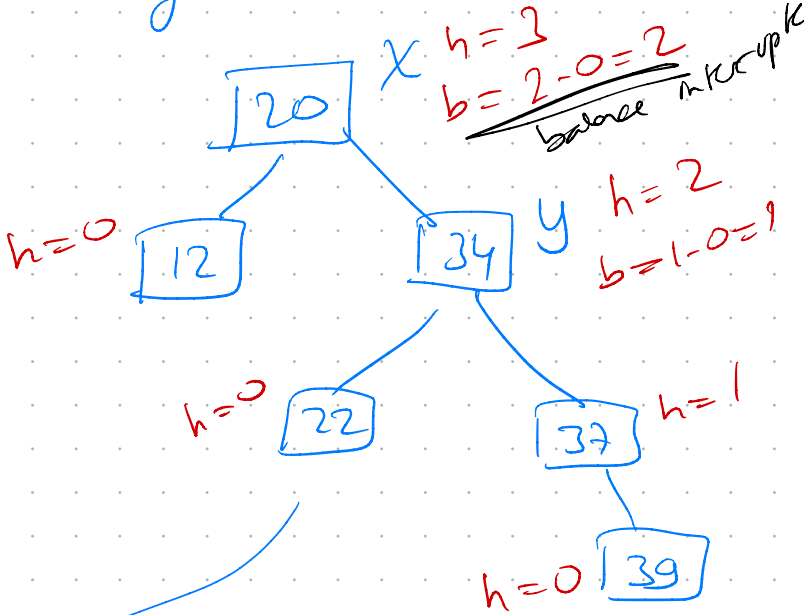
Self balancing trees

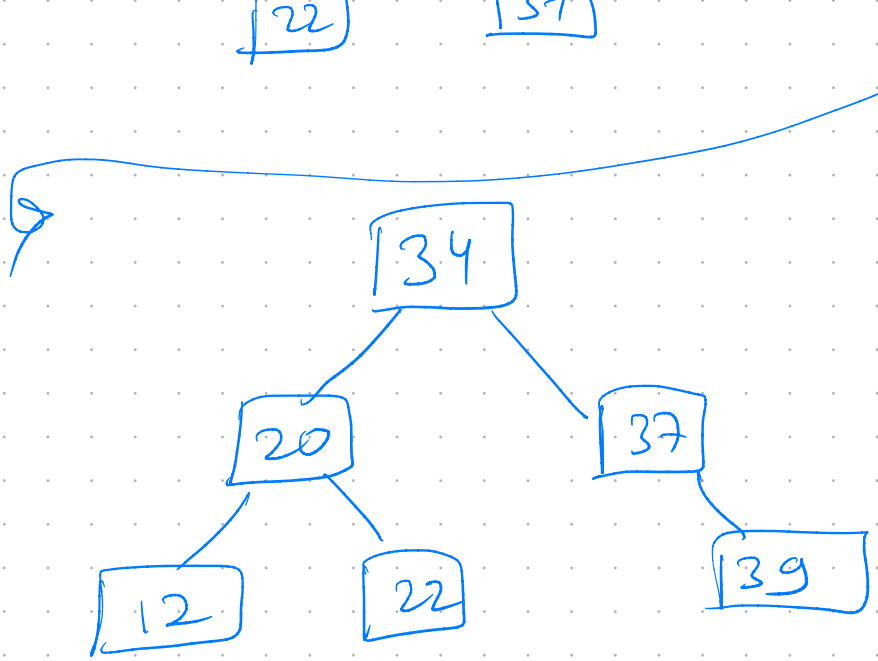
1) Why do we use self-balancing trees? Briefly explain the most important reason.

Runtime. Unbalanced worst-case can be O(n) while balanced search ... O(log n) time.

2) After inserting 39 into AVL tree, we obtain BST on right. How would you re-balance the right tree?







Hashing:

1) Insert the US Presidents listed below into a hash table w/ 13 cells using linear probing and the simple hash function h(key) = key.

Key	Name	Index
44	Obama	5
40	Reagen	1
39	Carter	0
31	Hoover	5
28	Wilson	2
26	Roosevelt	0
25	McKinley	12
24	Cleveland	11
17	Johnson	4
7	Jackson	7

$44 \% 13 = 5$

$40 \% 13 = 1$

$39 \% 13$

$31 \% 13$

$28 \% 13$

$26 \% 13$

$25 \% 13$

$17 \% 13$

0	Carter
1	Reagen
2	Wilson
3	Roosevelt
4	Johnson
5	Obama
6	Hoover
7	Jackson
8	
9	
10	Cleveland
11	
12	McKinley

$\frac{10}{13}$

2) What is the expected number of comparisons to find an item in this table?

Load factor $= \frac{10}{13} \approx 0.77$

$C = \frac{1}{2}\left(1 + \frac{1}{1-L}\right)$

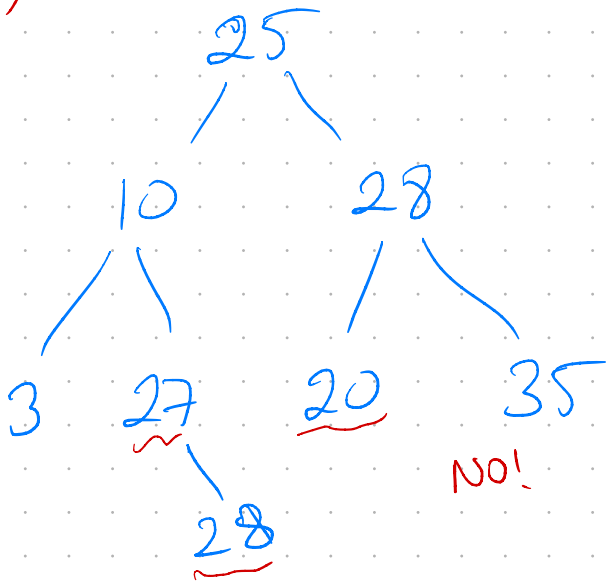
$C = \frac{1}{2}\left(1 + \frac{1}{\frac{3}{13}}\right) = \frac{1}{2}\left(1 + \frac{13}{3}\right)$

$= \frac{1}{2} \cdot \frac{16}{3}$

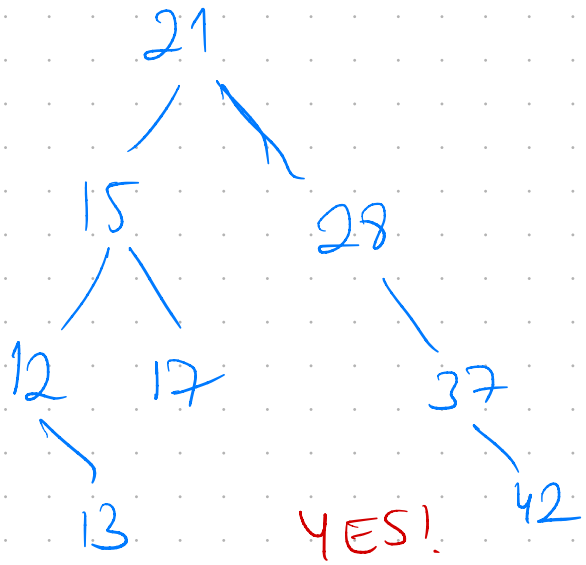
≈ 2.66

Binary Search Trees:

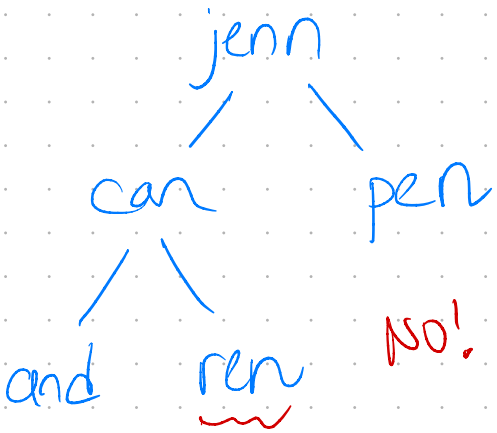
1) BST or not?



NO!

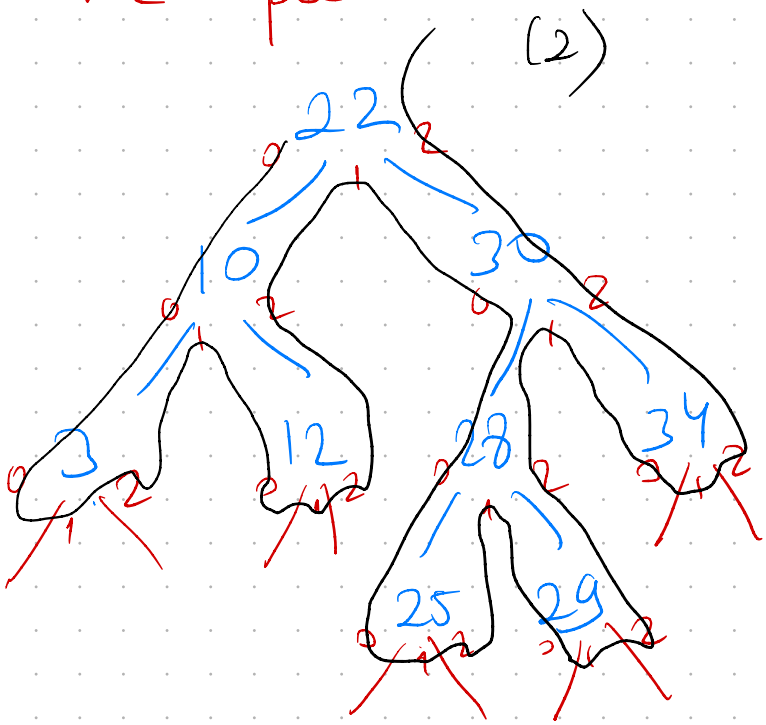


YES!



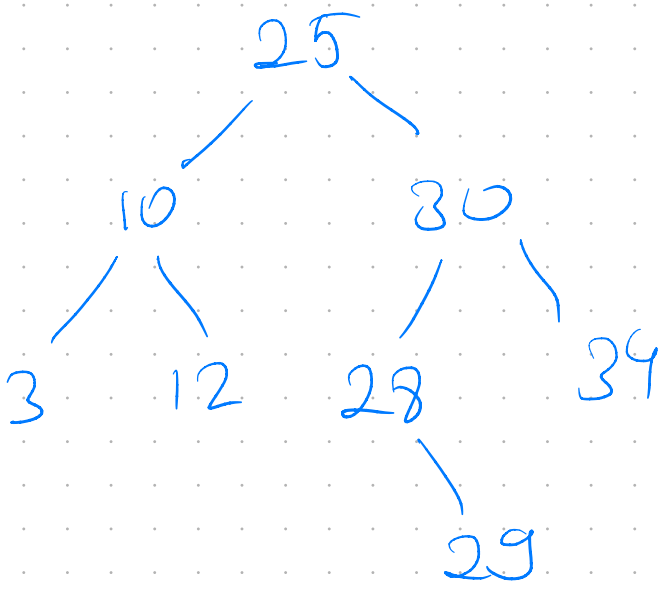
NO!

2) Find the post-order in the following BST.



3, 12, 10, 25, 29, 28, 34, 30, 22

3) Delete 22 from the previous BST.
you will change it w/ the successor of the node that you want to delete.
(min in the right subtree.)



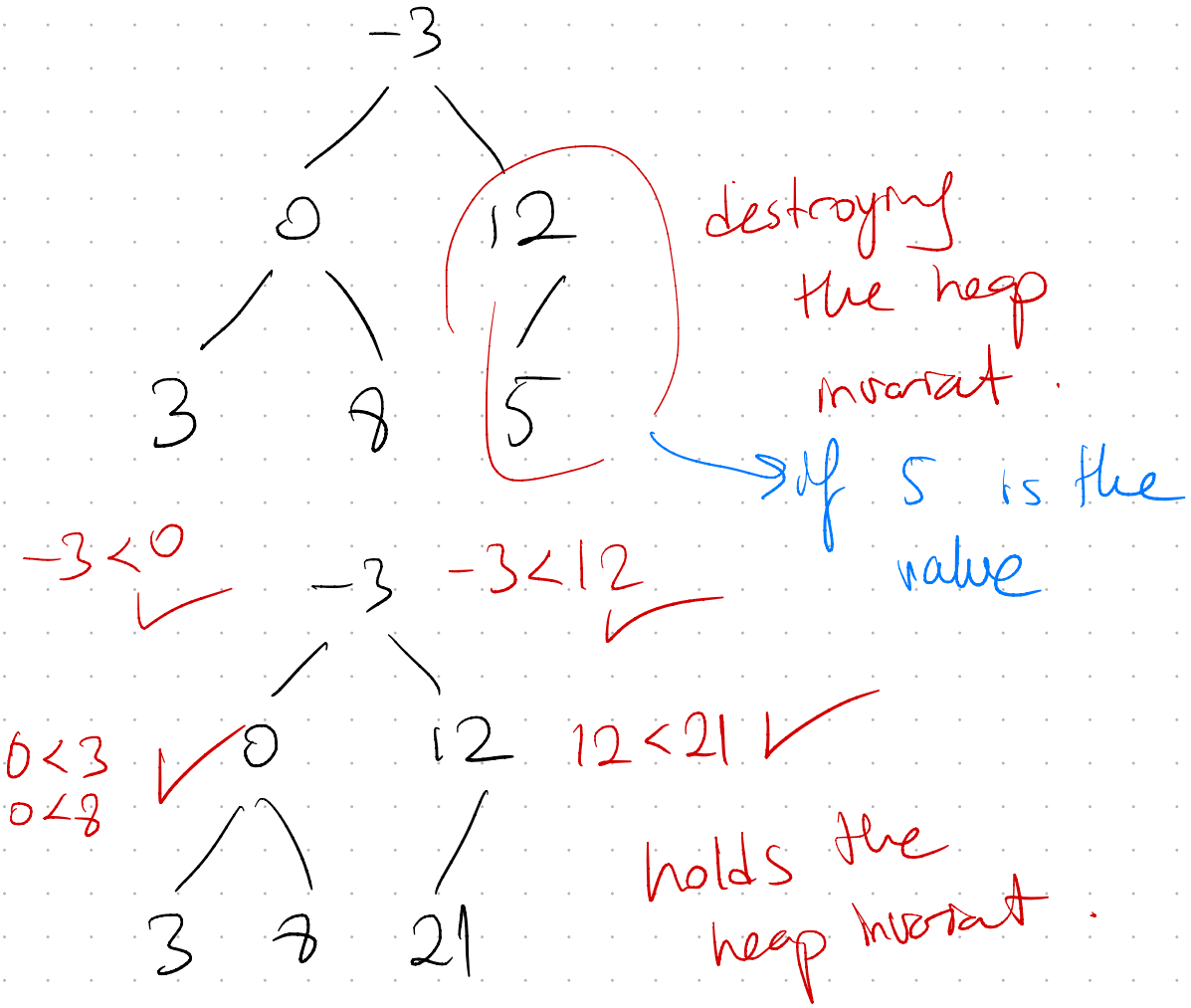
1) Write the following method public boolean isBalanced();
A binary tree is balanced if both of its subtrees are balanced & the height of its left subtree differs from the height of its right subtree by at most 1.

```
public boolean IsBalanced(){
    if(root==null)
      return true;
    return balanceOfNode(root);
}
public boolean balanceOfNode(Node n){
    int leftHeight, rightHeight;
    if( n==null)
      return true;
    leftHeight = height(n.left);
    rightHeight = height(n.right);
    if(Math.abs(leftHeight-rightHeight)<=1 &&
      isBalanced(n.left) && isBalanced(n.right))
      return True;
    return False;
}
```

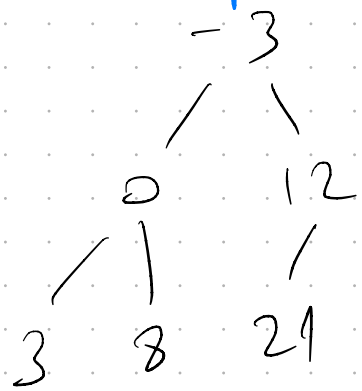
Heaps

1) Show whether the following array is a min heap or not. Justify your answer by drawing the tree representation and verifying the heap invariant is upheld.

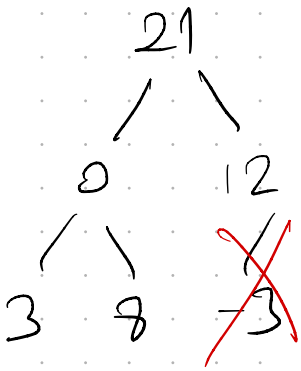
-3	0	12	3	8	21
p	l	r			

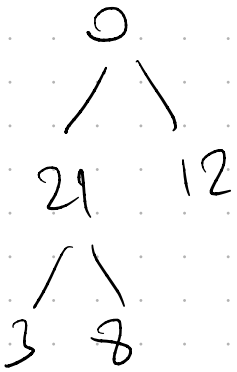


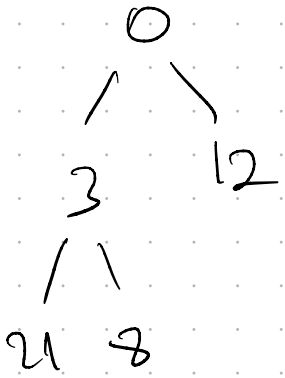
2) Apply heap removal operation twice & show the resulting heap after first & second removals, both in tree format and its array implementation. Show all node swaps.



remove -3

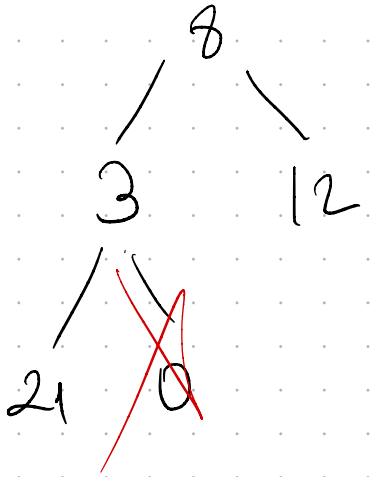


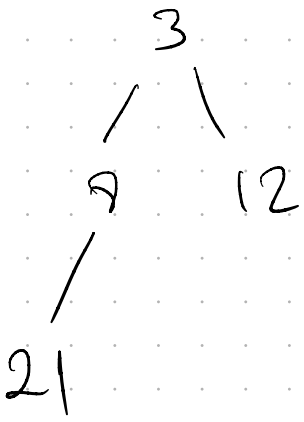




0	3	12	21	8

remove 0

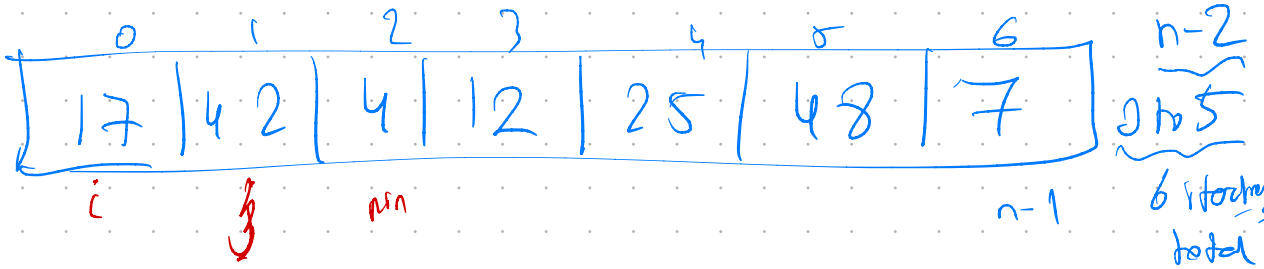




3	8	12	21

Sorting

1) Use selectionSort to sort the following values.



1st Iteration: 4 42 17 12 25 48 7 (min)

2nd Iteration: 4 7 17 12 25 48 42 (min)

3rd Iteration: 4 7 12 17 25 48 42

4th Iteration: 4 7 12 17 25 48 42

5th Iteration: 4 7 12 17 25 48 42

6th Iteration: 4 7 12 17 25 42 48

1) Use mergesort to sort the following values