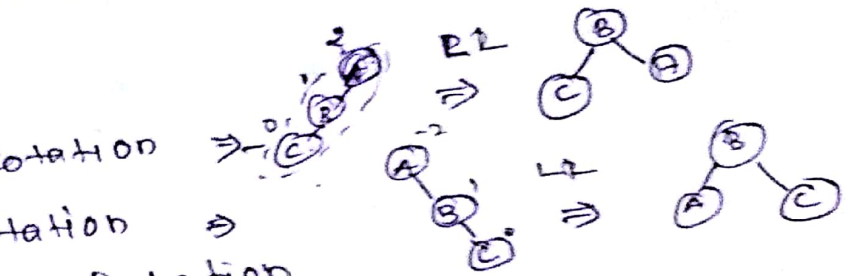
AVL Tree

- Self balancing binary search tree ∵ AVL Tree
  Adelson, Velsky, Landis. (AVL)

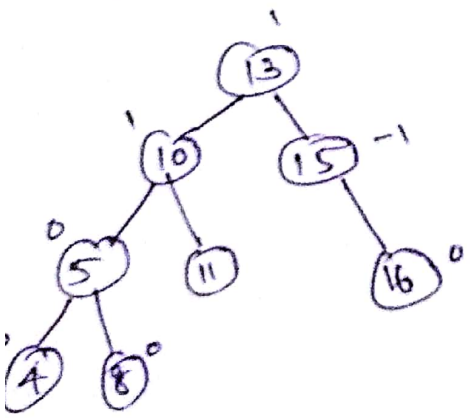
- Height of a node.
- Height of tree.
- Balancing factor
  = Height (left subtree) − Height (Right subtree)

- Balance factor must be −1, 0, 1
  but not more than (−1, 0, 1)

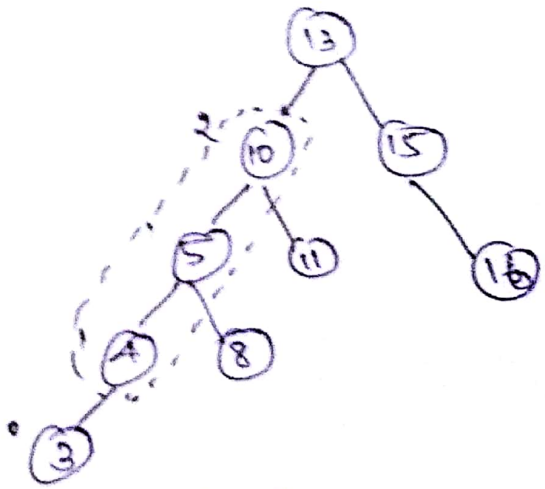
- 4 - Rotation
  - Right Rotation ⇒
  - Left Rotation ⇒
  - Left-Right Rotation
  - Right-Left Rotation



Right Rotation:

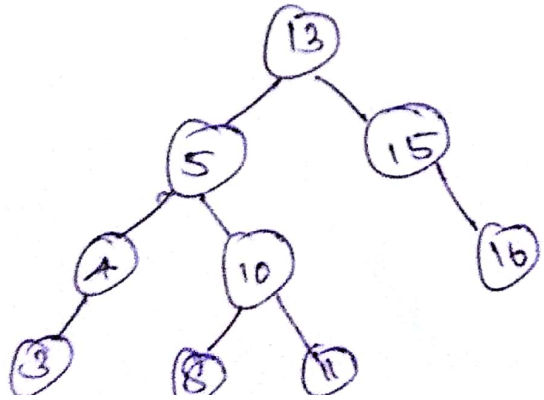


insert 3

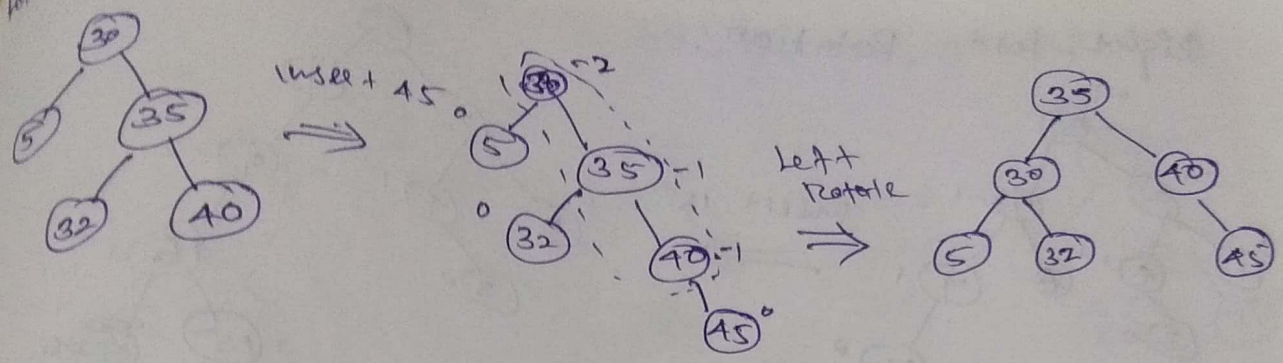


imbalance.

⇒

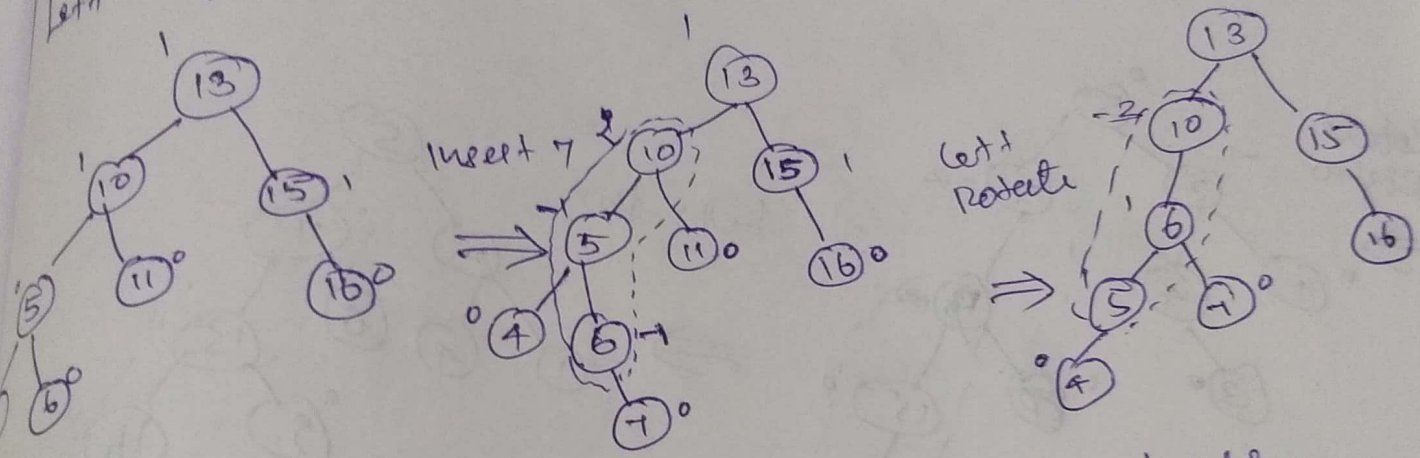


Left Rotation



Imbalance

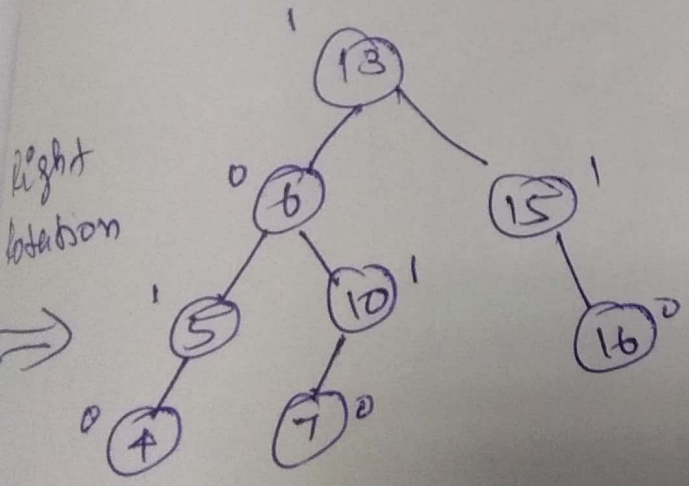
Left-Right Rotation:



Imbalanced.

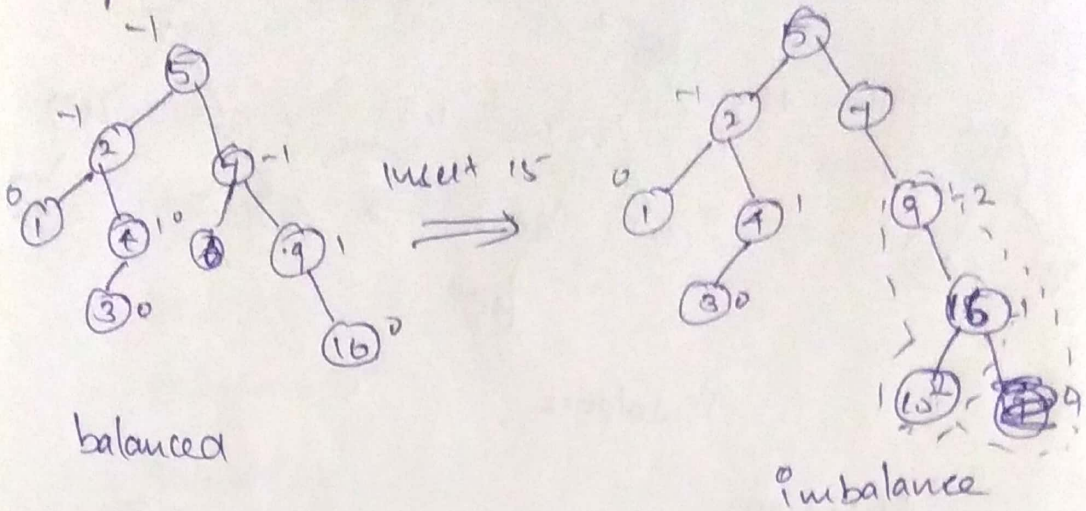
Imbalance.

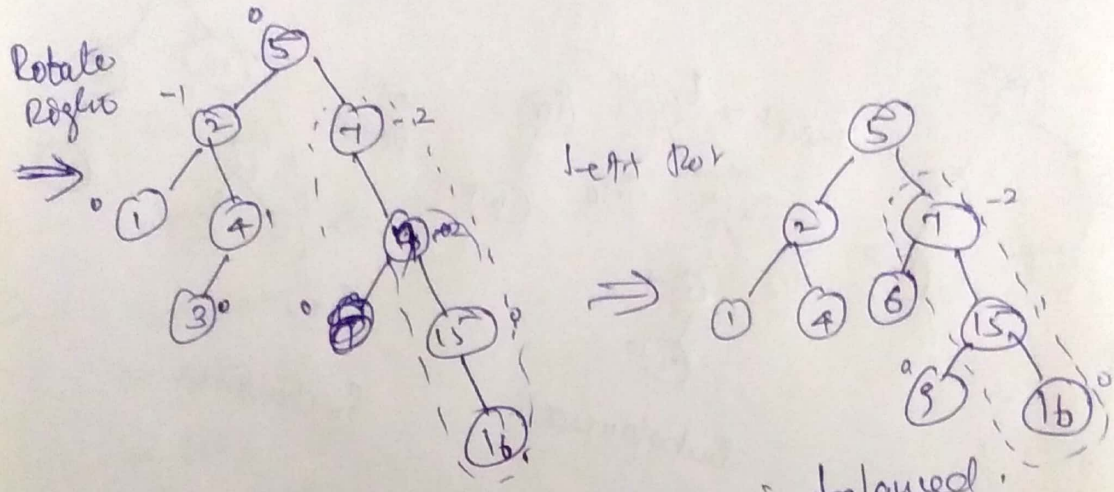
Right Rotation

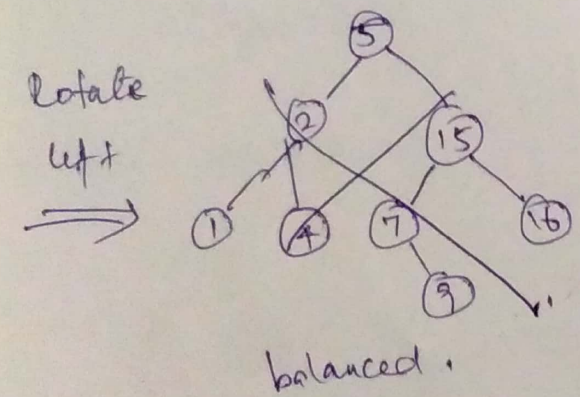


balanced.

Right - left Rotation:



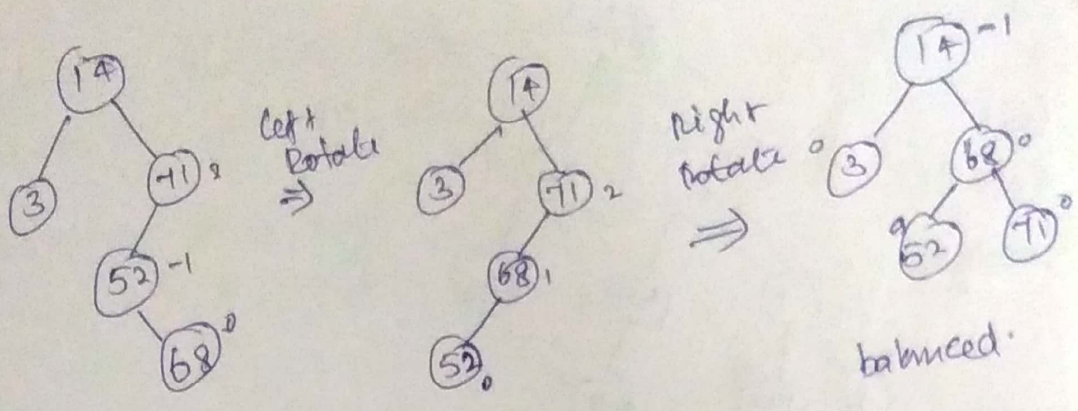


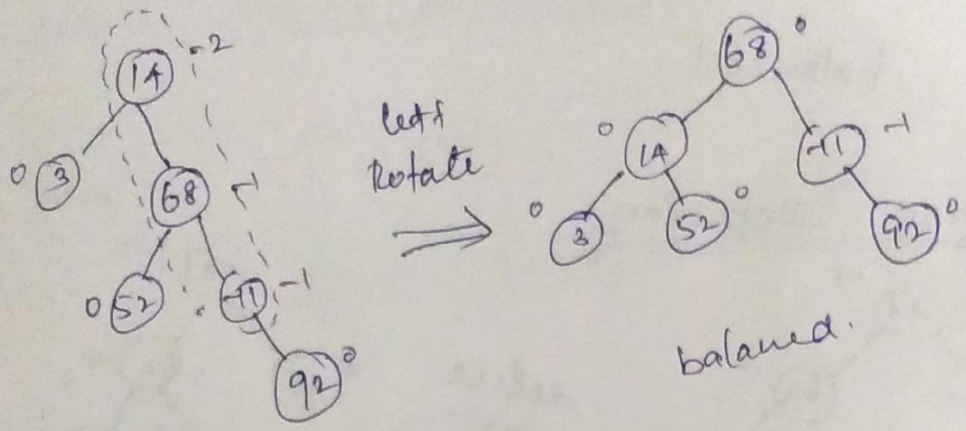


Right subtree of left child ⇒ left Rotate

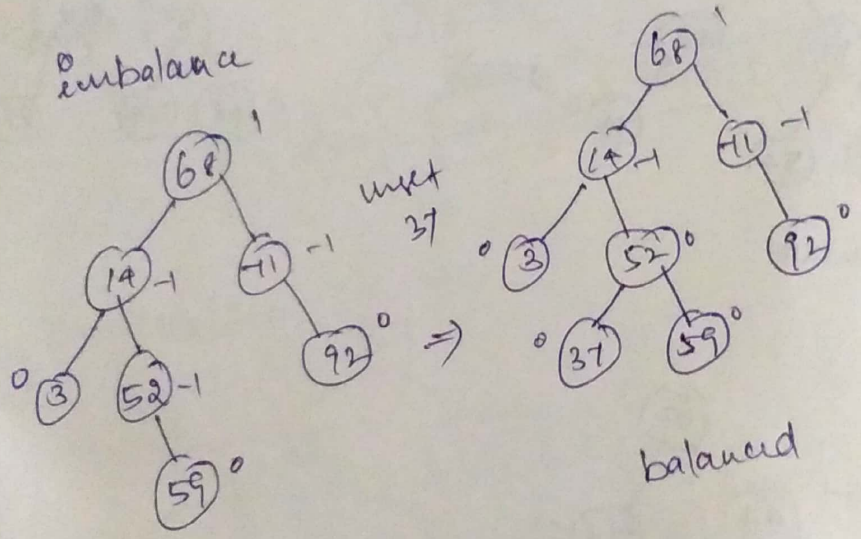
left subtree of right child ⇒ Right Rotate

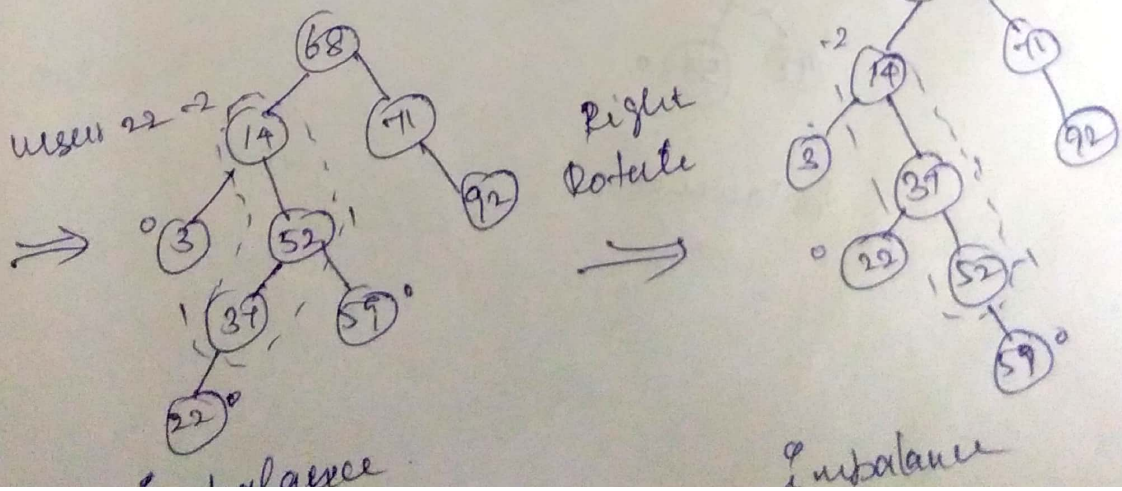
Example: 14, 71, 3, 52, 68, 92, 59, 37, 22

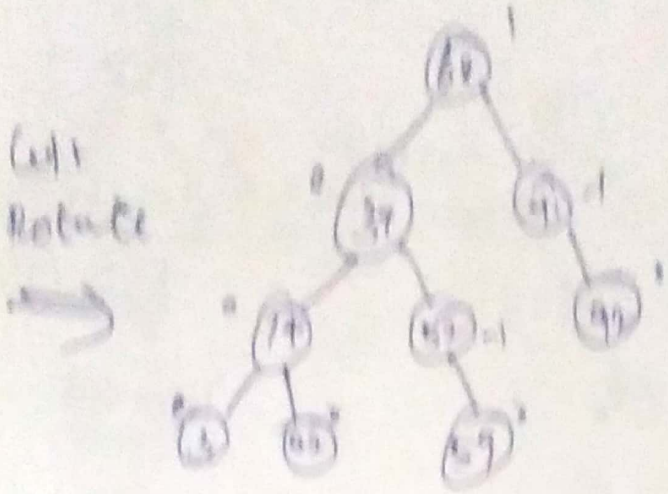




imbalance

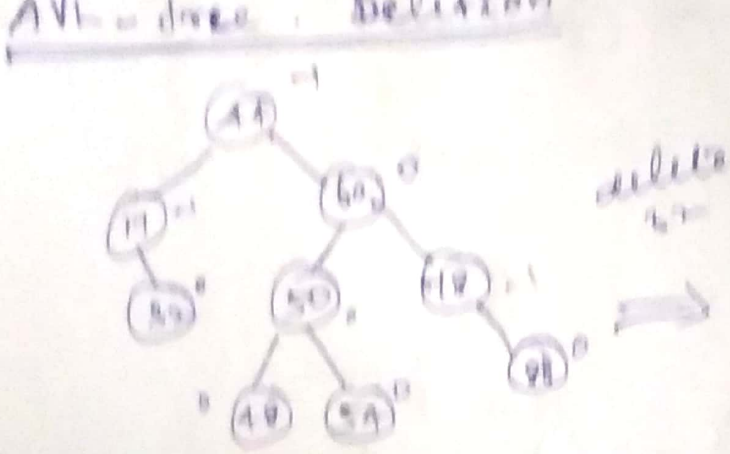


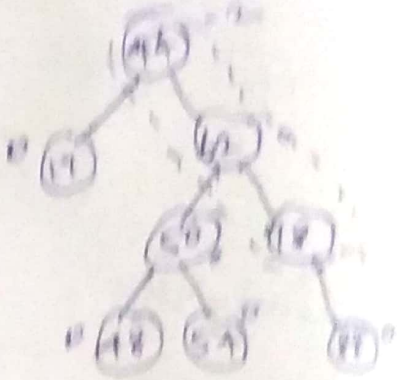




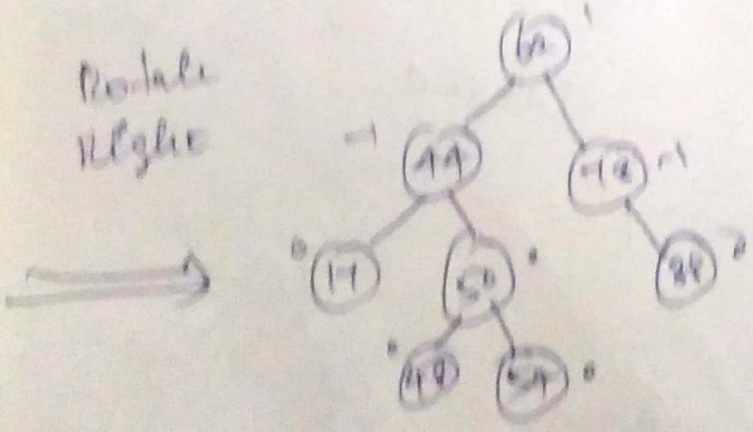
balanced.

AVL - tree : Deletion





balanced



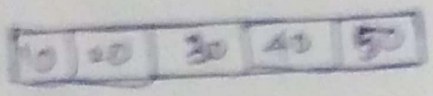
balanced.

B-Tree

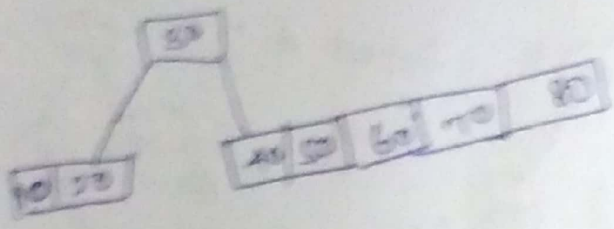
Construct B-tree with elements
10, 20, 30, 40, 50, 60, 70, 80, 90. min degree t = 3

✓ maximum no. of keys = 2t-1 = 5
in a node

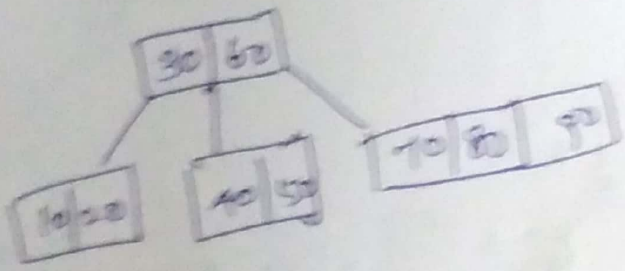
Step 1:



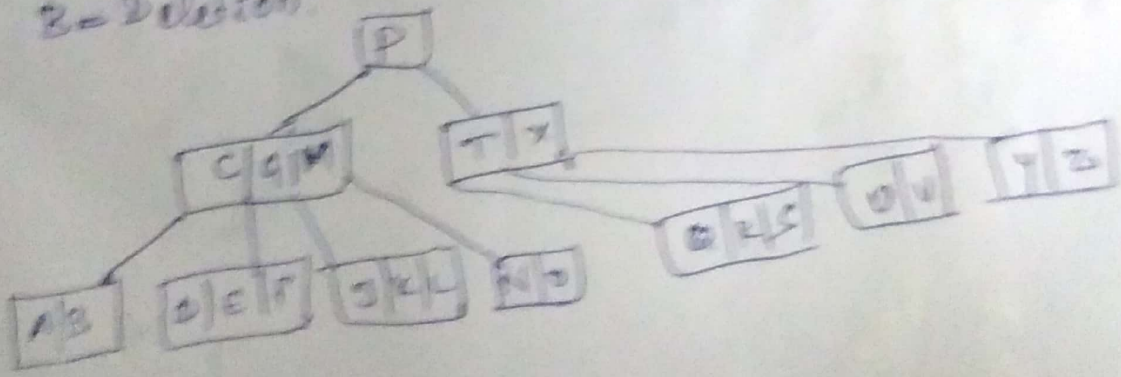
Step 2:



Step 3:



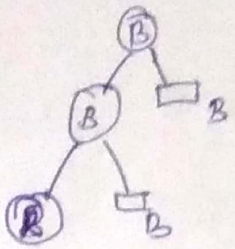
3- Deletion



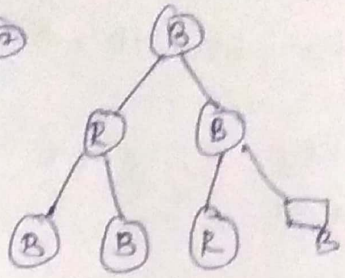
Delete F:

RED. BLACK TREE

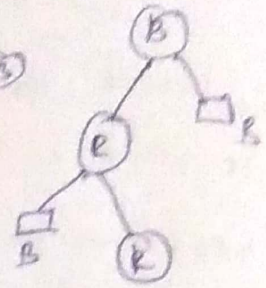
①



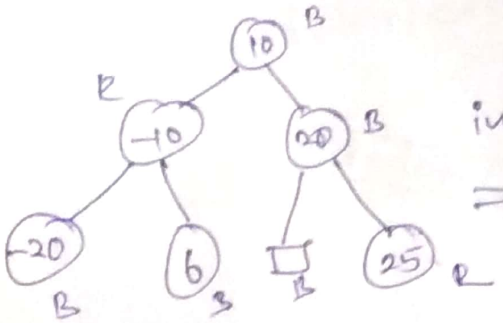
②



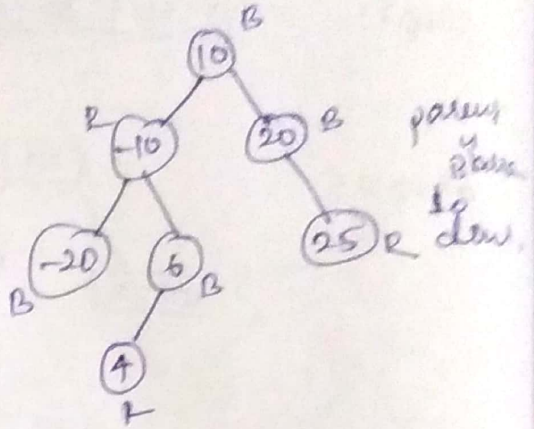
③



Insertion:

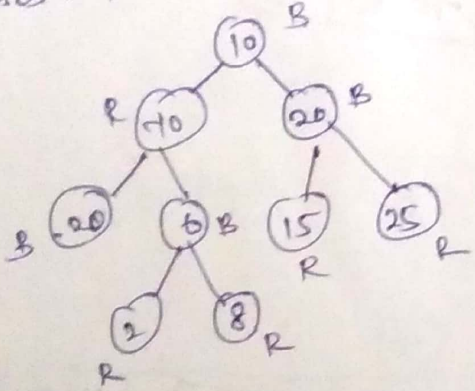


insert 4 ⟹

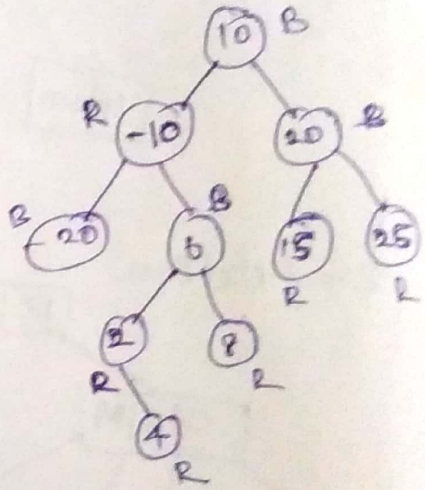


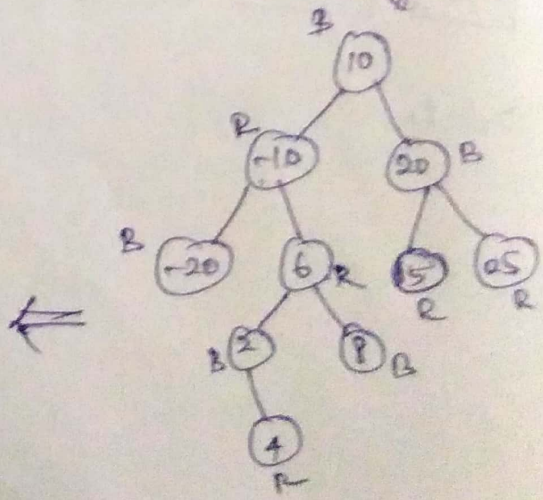
parent is Black so done.

Cases 2 :

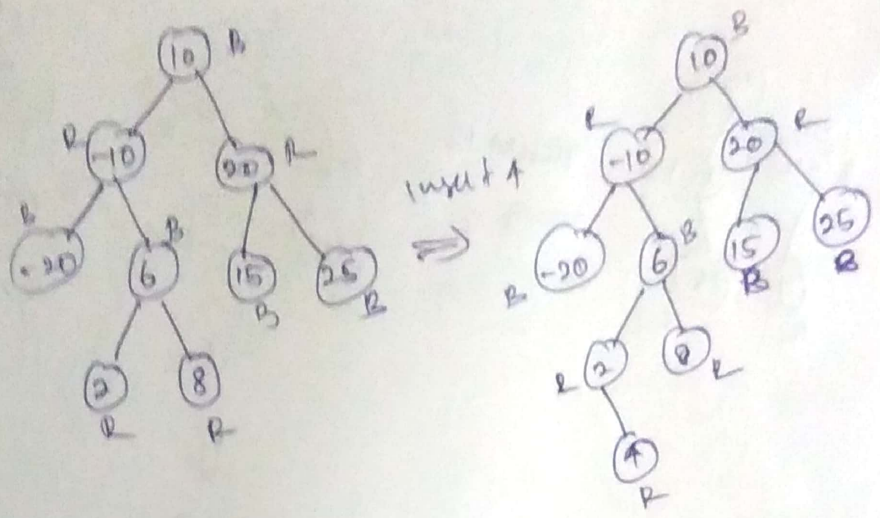


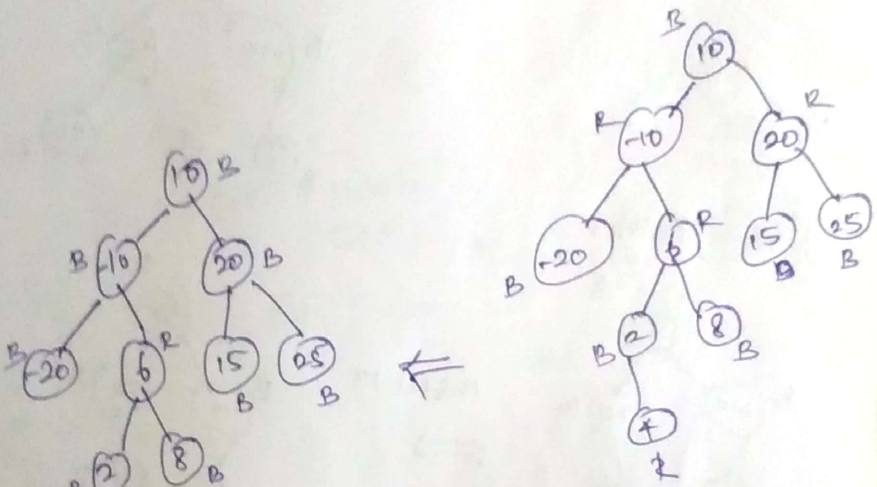
insert 4 ⟹





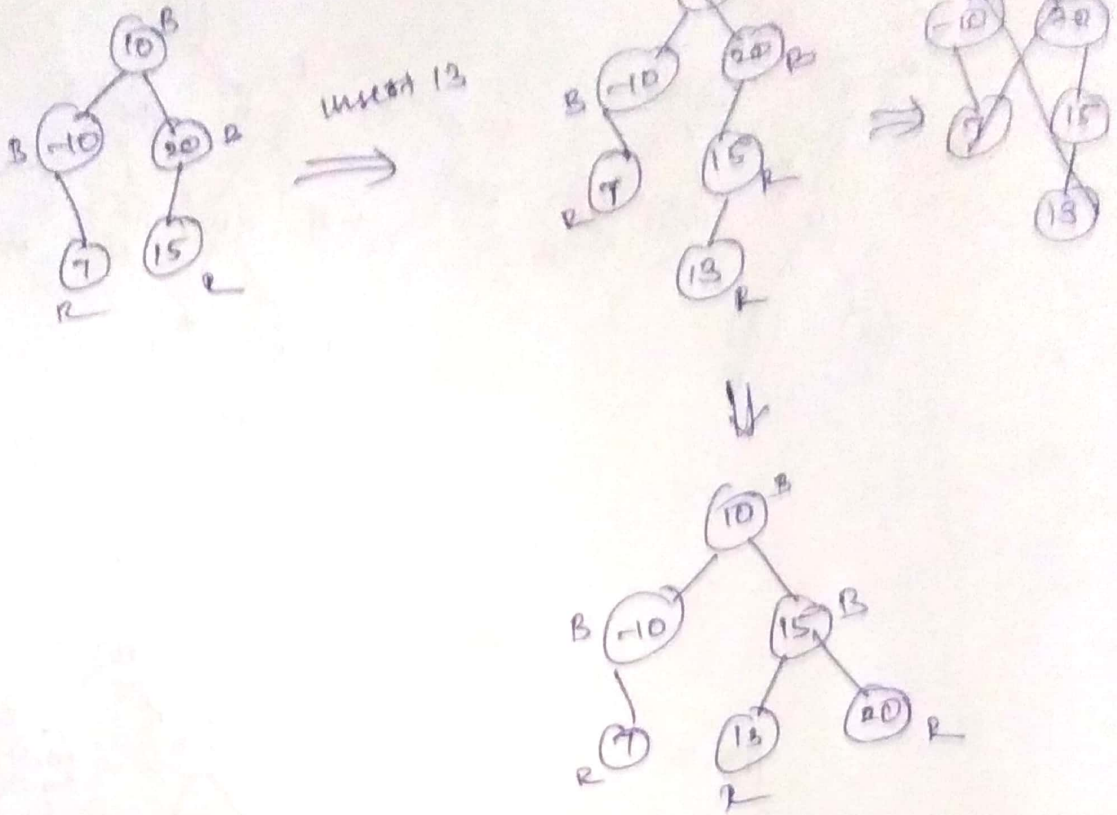
Case 3:





red black tree

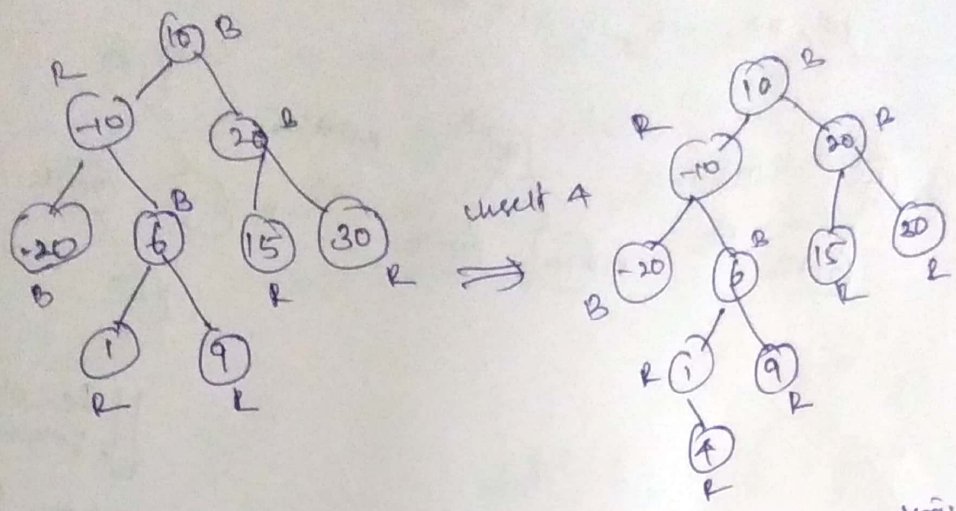
Insertion with Rotation:



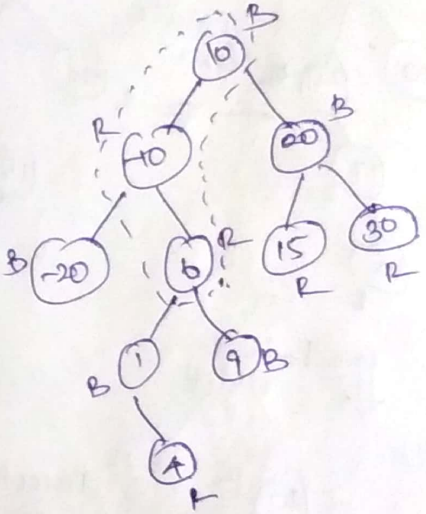
L-R Rotation:

R-B tree.

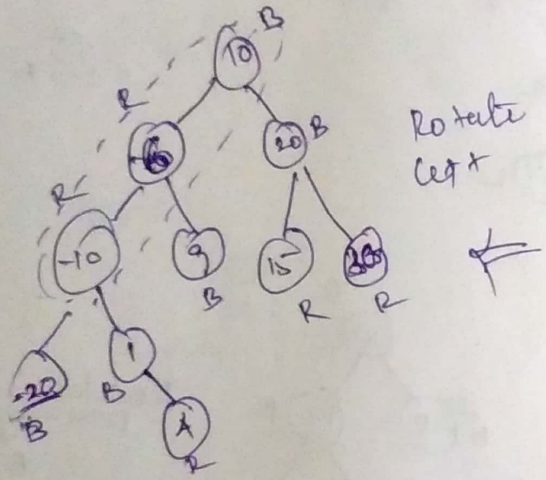
parent sibling is Black:



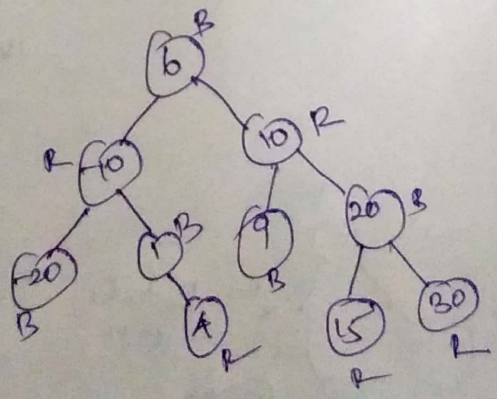
recoloring.



Rotate left



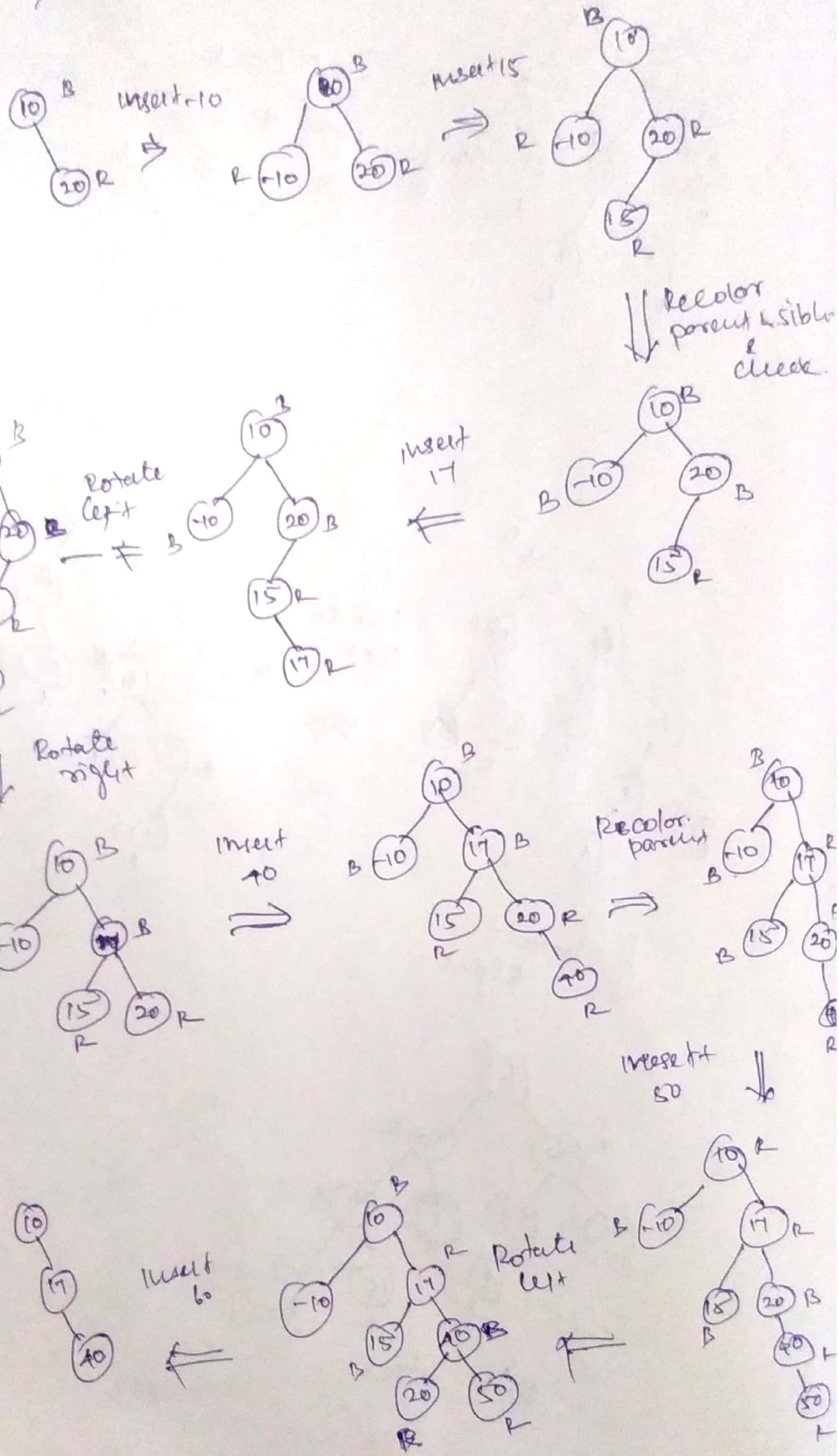
rotate right.



R-D. free.

Construct Red-Black tree:

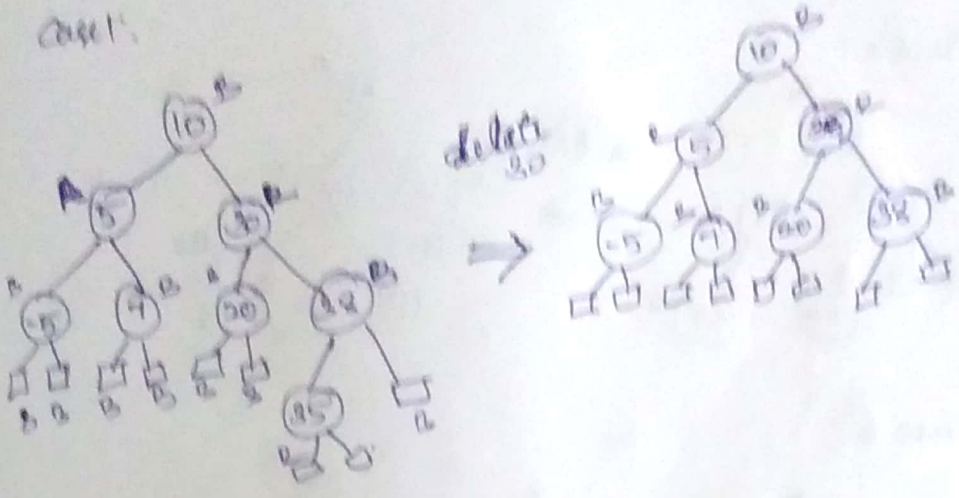
10, 20, -10, 15, 17, 40, 50, 60.



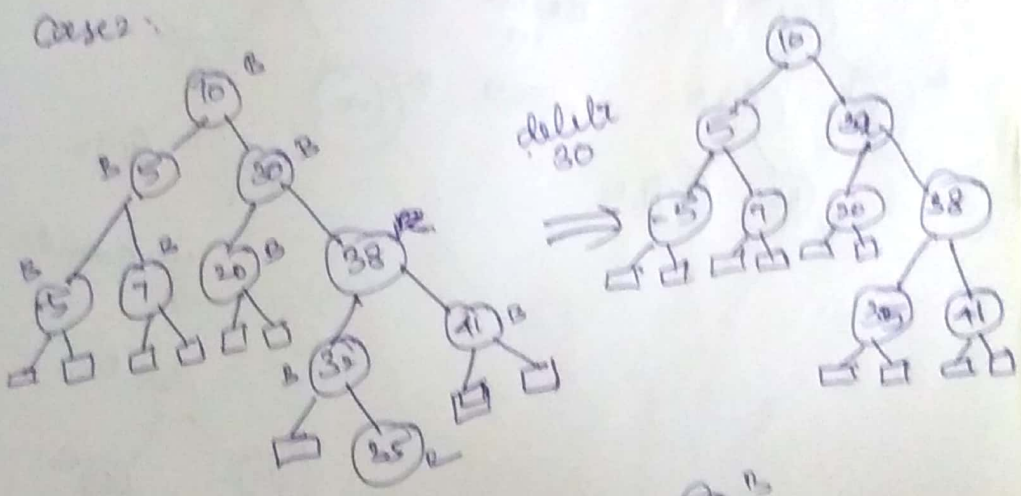
Red Black tree Deletion:

- convert it to a 0 or 1 child case
- If red node just delete it.
- If black node with red child then delete black & turn red into black node.

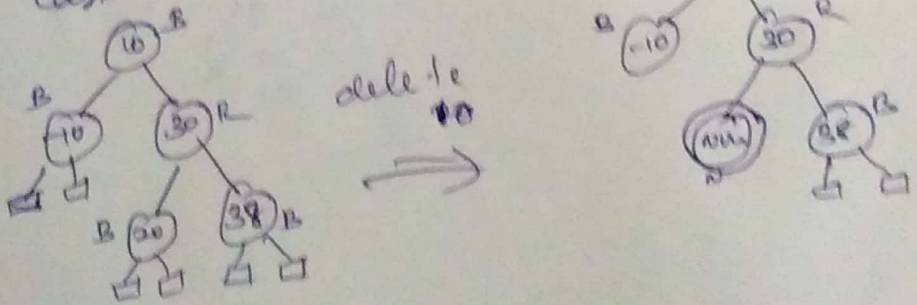
Case1:

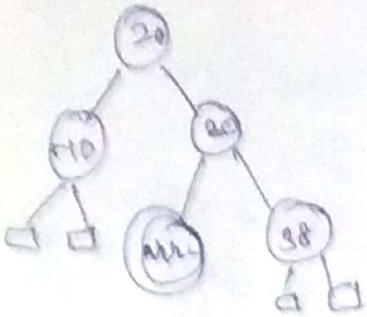


Case2:

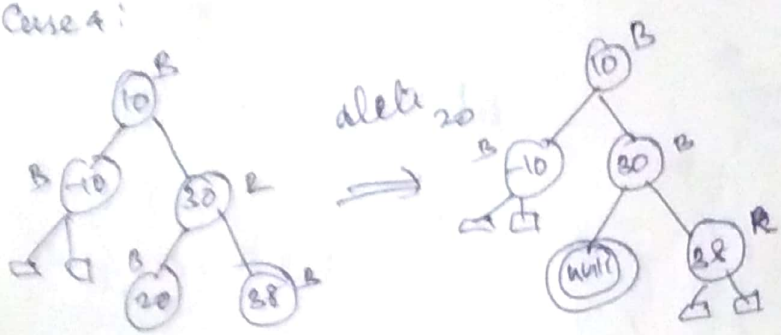


Case 3:

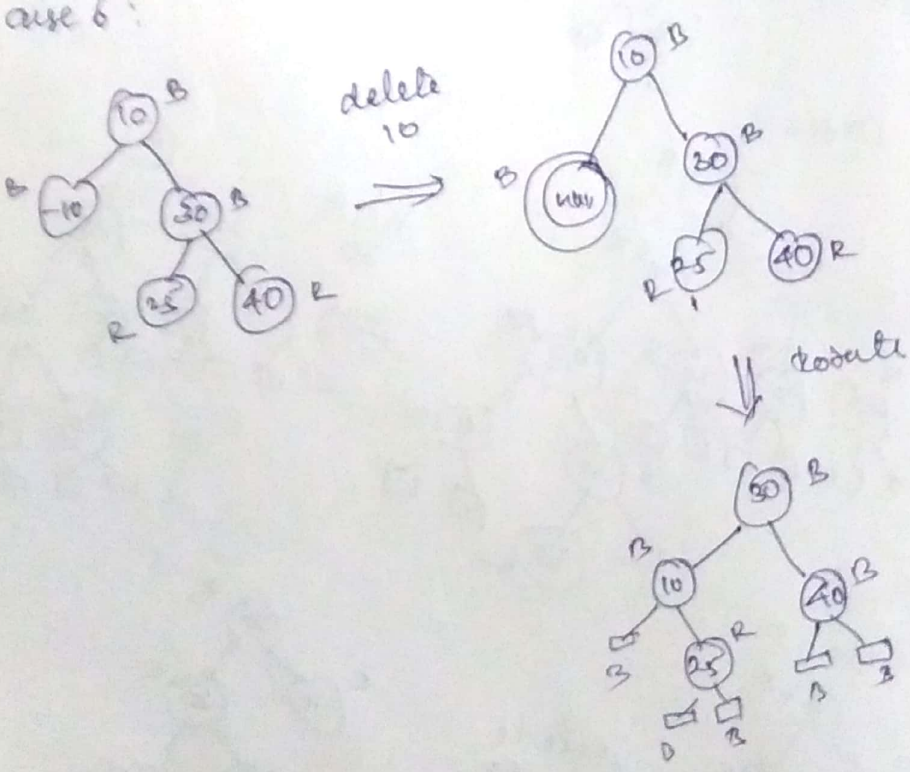




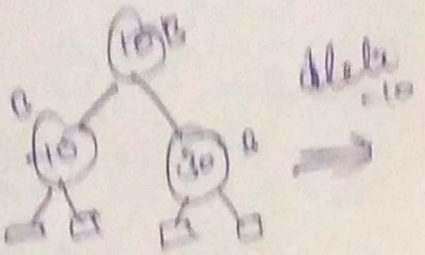
Case 4 :

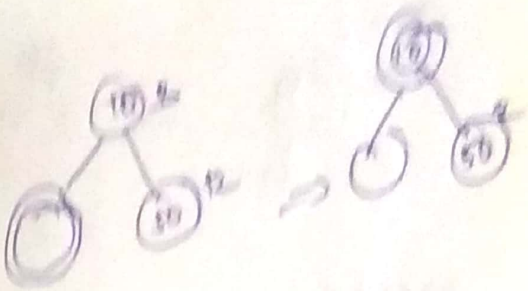


case 6 :



Case 4





Case 5