Red-Black Tree —

A Red black Tree is a binary Search tree with one extra bit of storage per node, its color which can be either Red or Black. By constraining the way nodes can be colored on any path from root to a leaf, red black tree ensures that no such path is more than twice as long as any other, so that the tree is approximately balanced.

→ Although the balance of the tree is not perfect, it is good enough to reduce the searching time and maintain it around $O(\log n)$, where n is the total no of elements in the tree.

→ It was invented by Rudolf Bayer in 1972.

→ Each node of the tree now contains fields color, key, left right and p

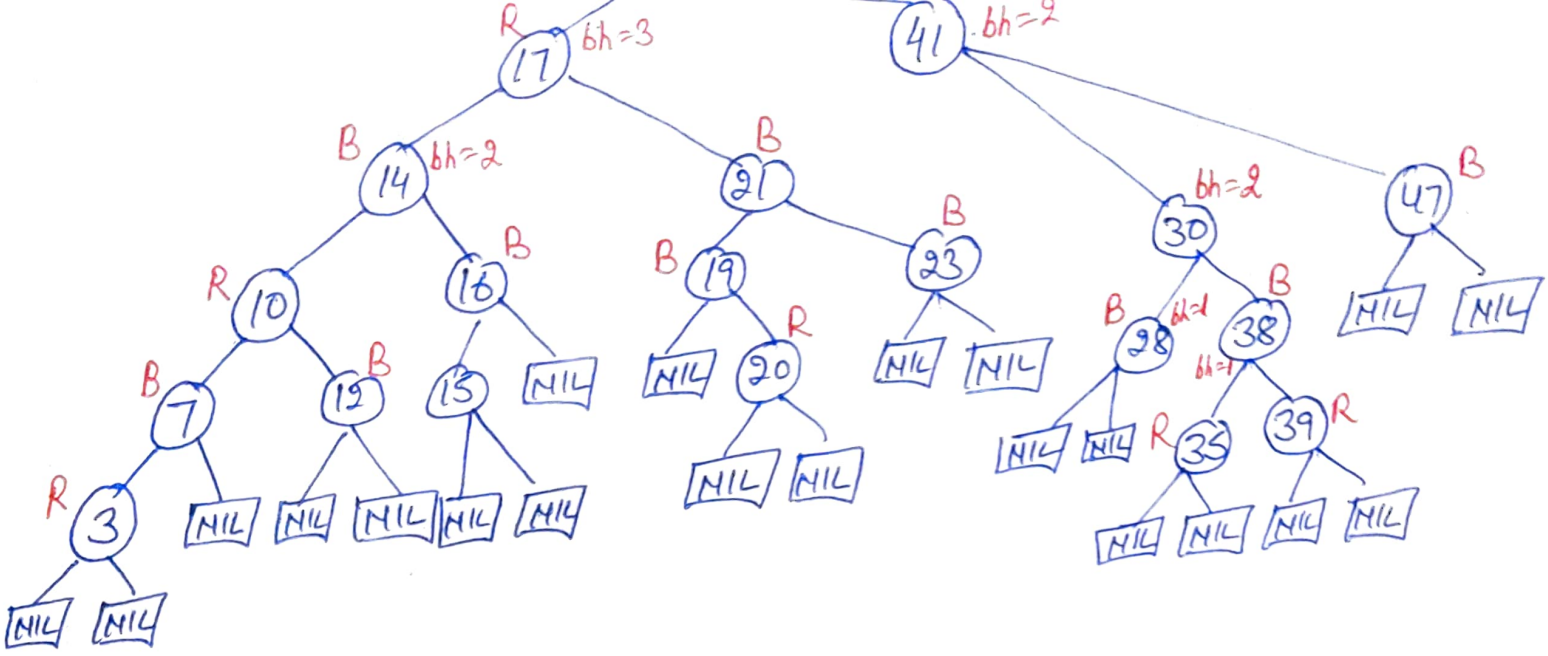
A Binary Search Tree is a Red Black tree if it satisfies the following properties:

1. Every node is either red or black
2. Root is always Black
3. If a node is Red then both its childrens are Black
4. Every simple Path from a node to a descendant leaf contains the same number of black nodes.

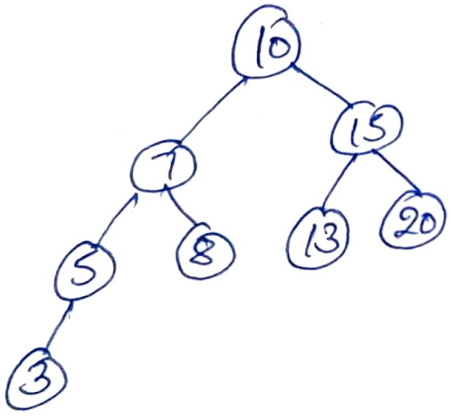
Black Height of a Node —

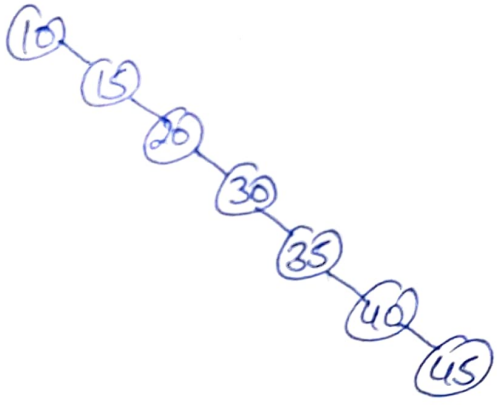
We call the number of black nodes on any path from, but not including, a node x to a leaf the black height of the node, denoted $bh(x)$.

We define the black height of a tree to be the black height of root node.



e.g Red Black Tree





Right skewed BST

Insertion Into Red-Black Tree —

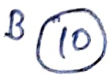
1. If tree is empty, create new node as root node with color black.
2. If tree is not empty, create new node as leaf node with color Red.
3. If Parent of new node is black then exit.
4. If Parent of new node is Red then check the color of Parent's Sibling of new node
   a) If color is black or NULL then do suitable rotation and recolor
   b) If color is Red then recolor and also check if Parent's Parent of new node is not root node, then ~~recle~~ recolor and recheck.

Properties of RB Tree
- root = Black
- no red-red relationship
- same no of Black node in every path

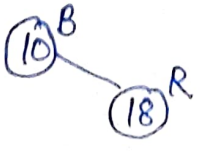
Insert In Red-Black Tree →

10, 18, 7, 15, 16, 30, 25, 40, 60, 2, 1, 70

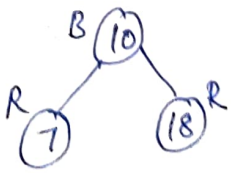
① Insert 10



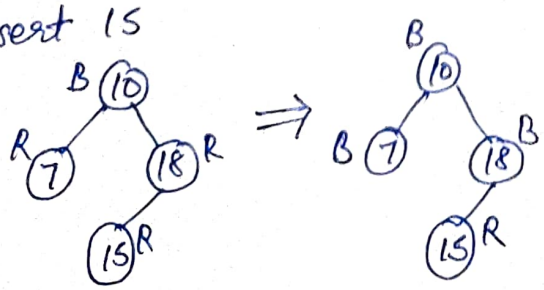
② Insert 18



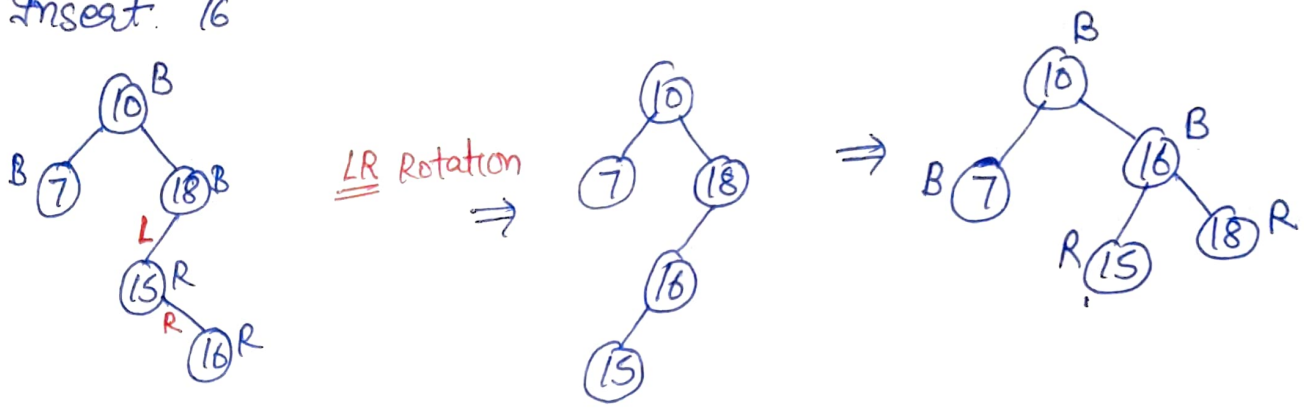
③ Insert 7



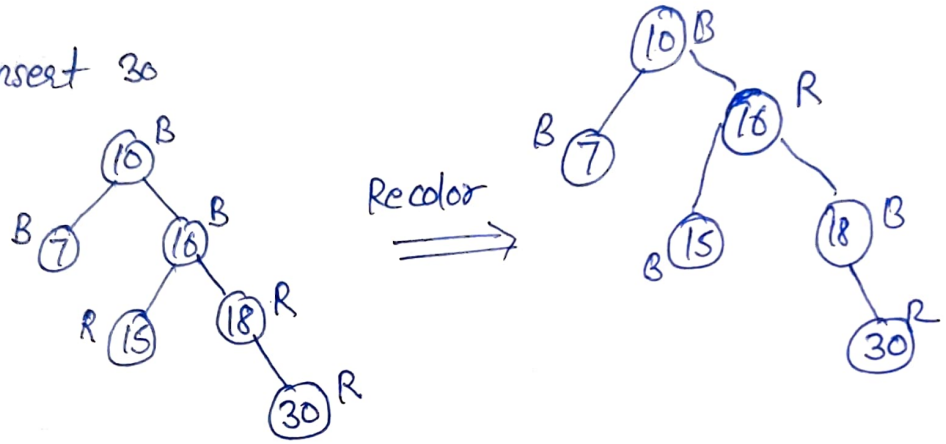
④ Insert 15



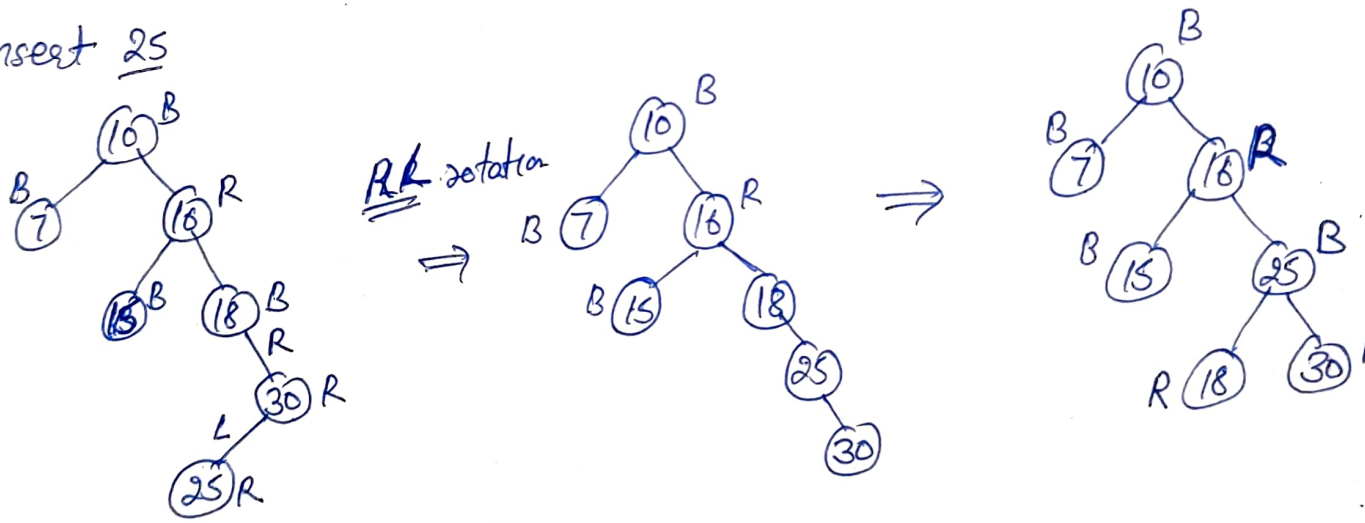
⑤ Insert 16



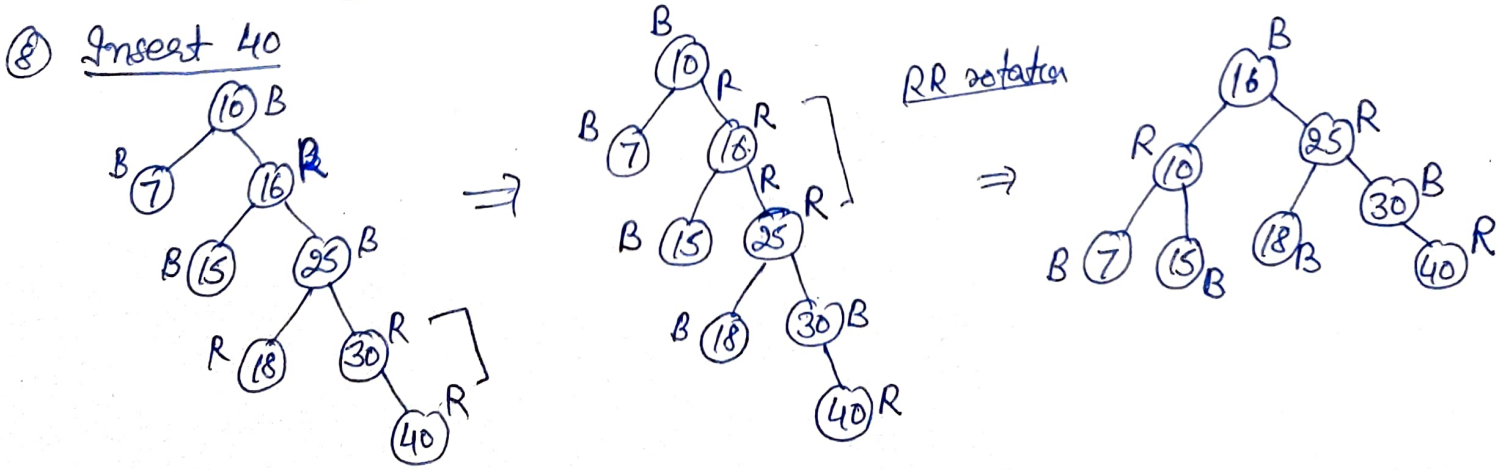
⑥ Insert 30



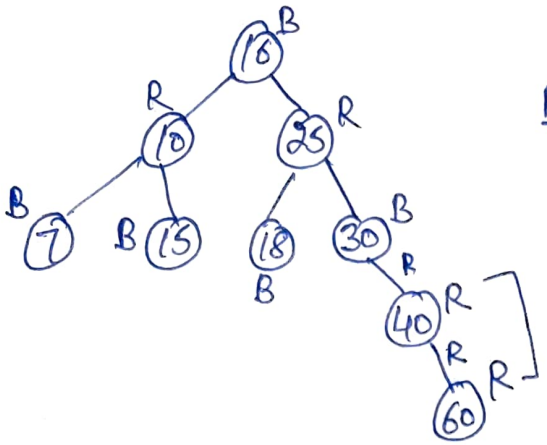
⑦ Insert 25



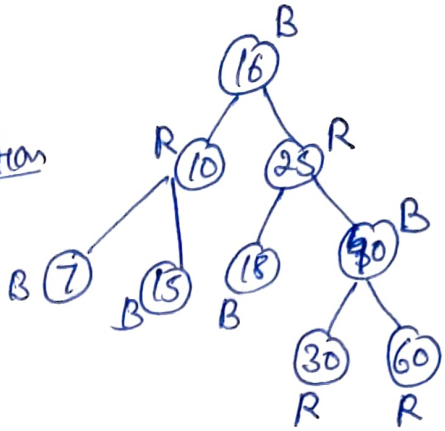
⑧ Insert 40



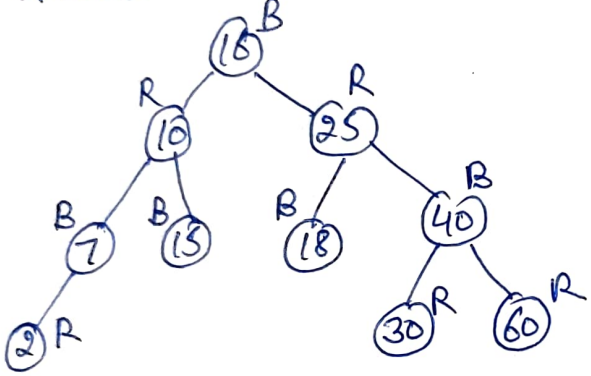
⑨ Insert 60



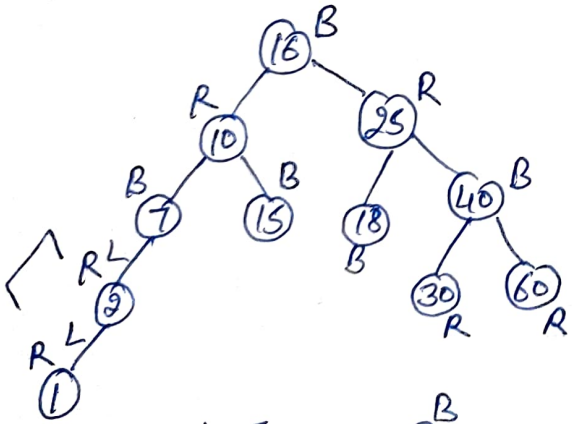
RR rotation



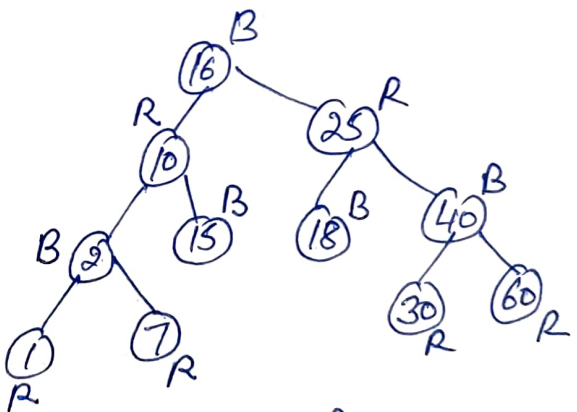
⑩ Insert 2



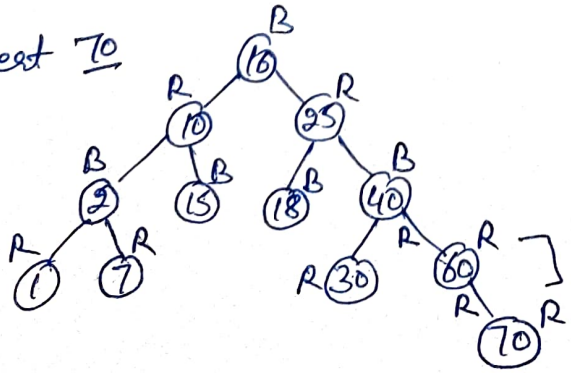
⑪ Insert 1



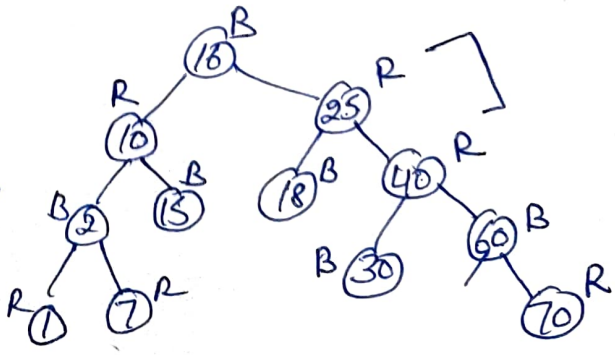
LL

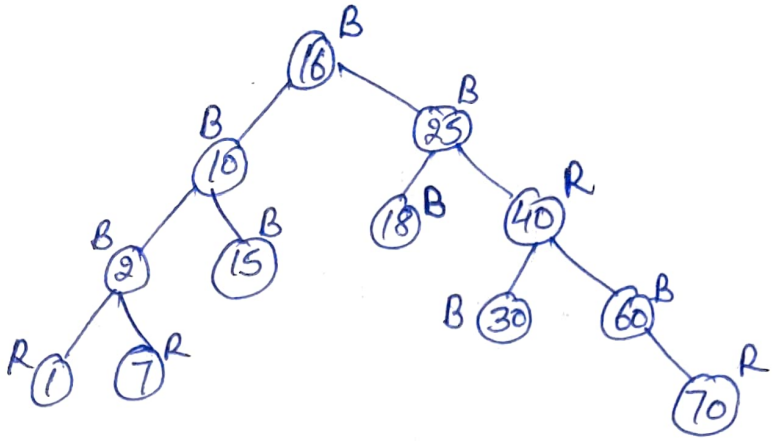


⑫ Insert 70



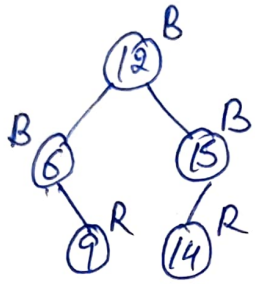
⇒



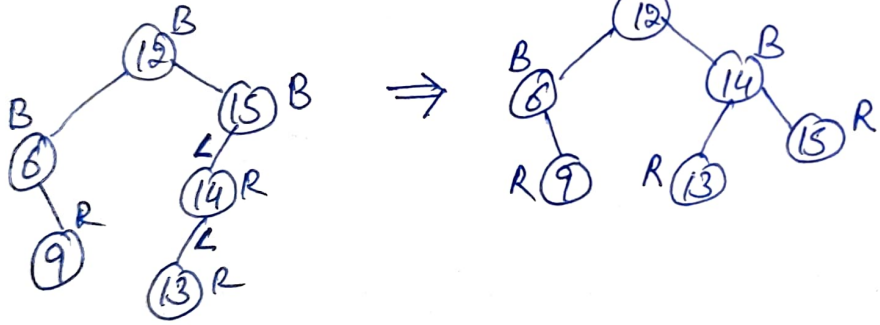


Ans

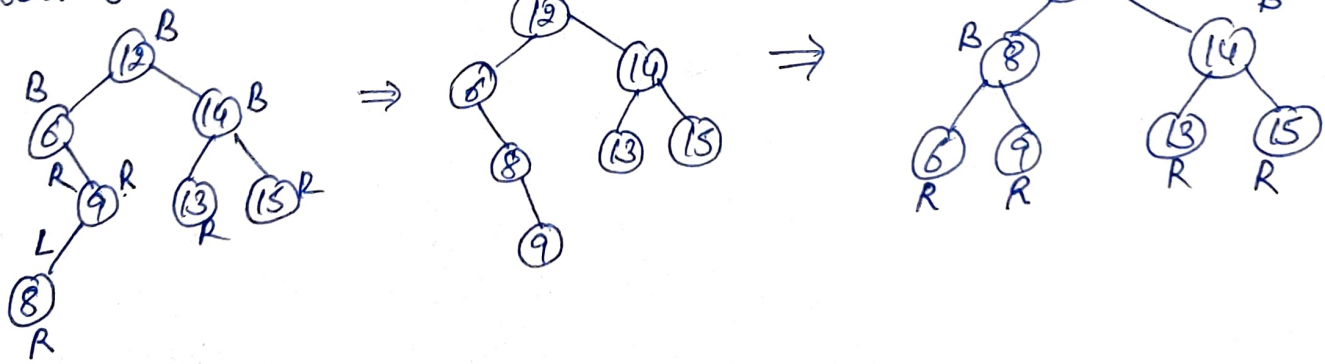
e.g. Insert into RB Tree



① Insert 13



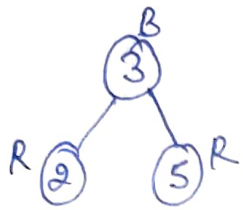
② Insert 8



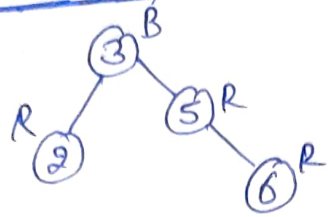
Q Insert the following into RB-Tree

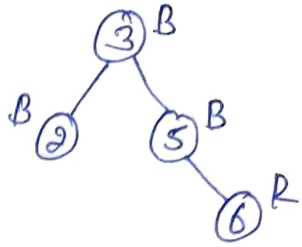
3, 2, 5, 6, 10, 4, 8, 9

Sol^{n}

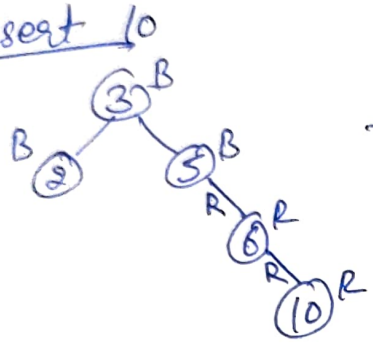


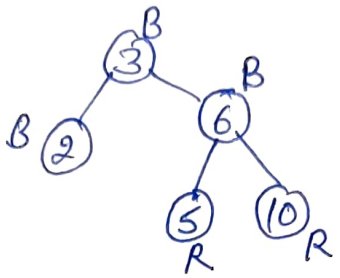
Insert 6



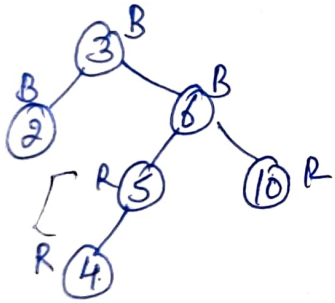


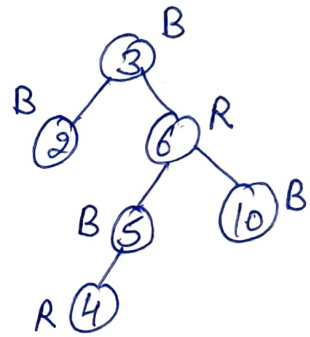
Insert 10



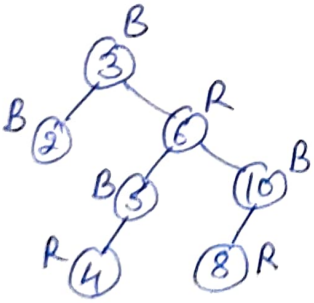


Insert 4

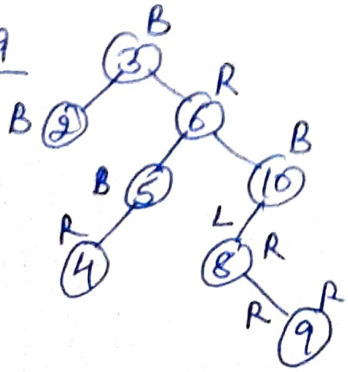


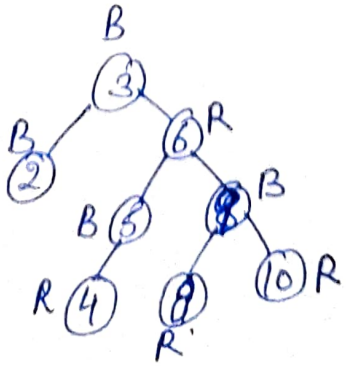


Insert 8



Insert 9



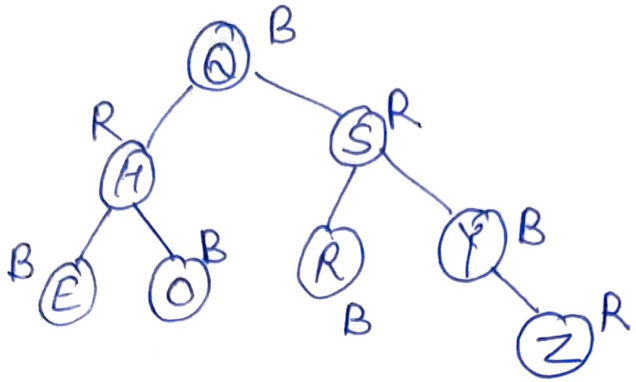


Ans

Q Insert into RB Tree

H, R, E, O, Q, S, Y, Z

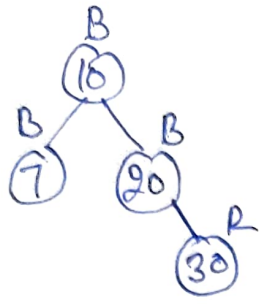
Ans



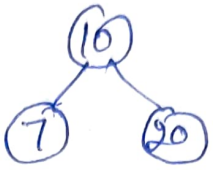
step 1 Perform BST deletion.

step 2: case 1 : If node to be deleted is Red, just delete it

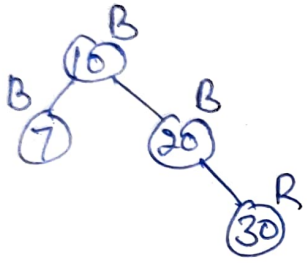
e.g 1



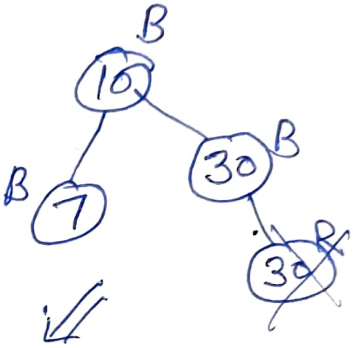
Delete 30 →

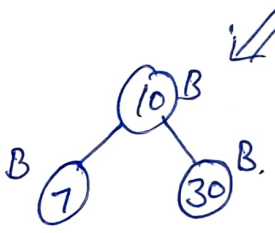


e.g 2

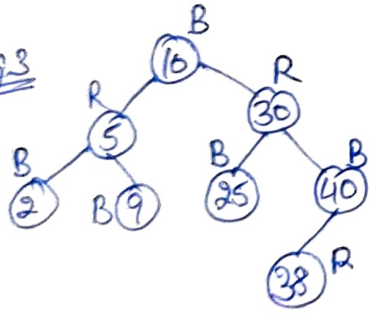


Delete 20 with 30
Replace 20 ~~with its inorder succ~~

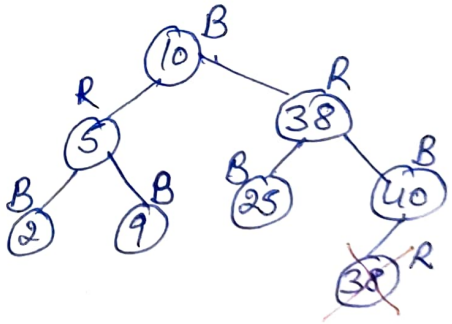




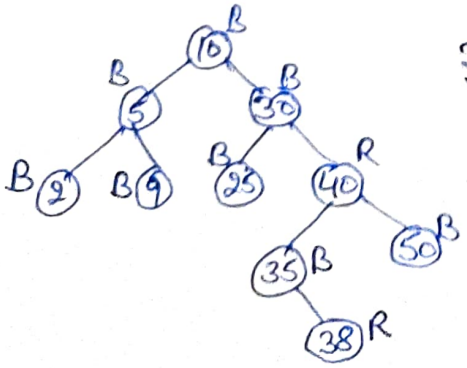
e.g 3



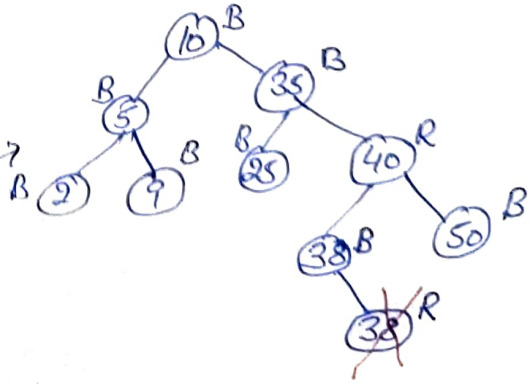
delete 30
replace 30 with its inorder successor



e.g 4



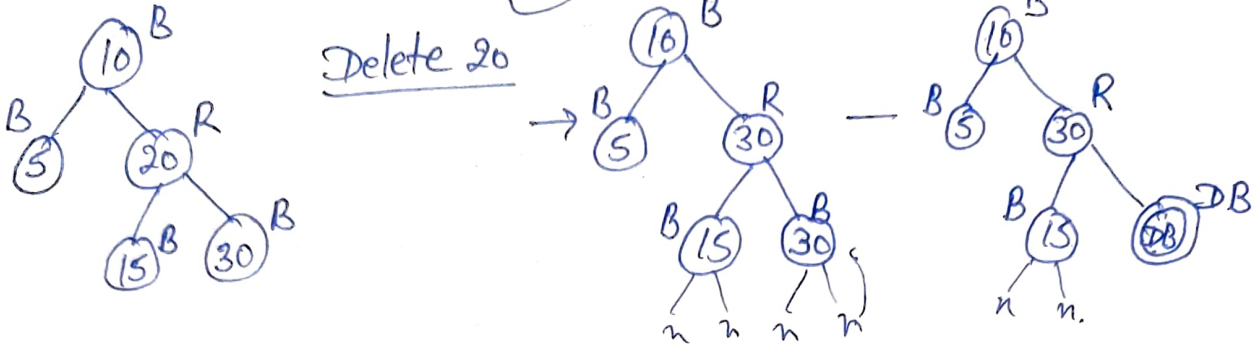
Delete 30
Replace with its inorder succ

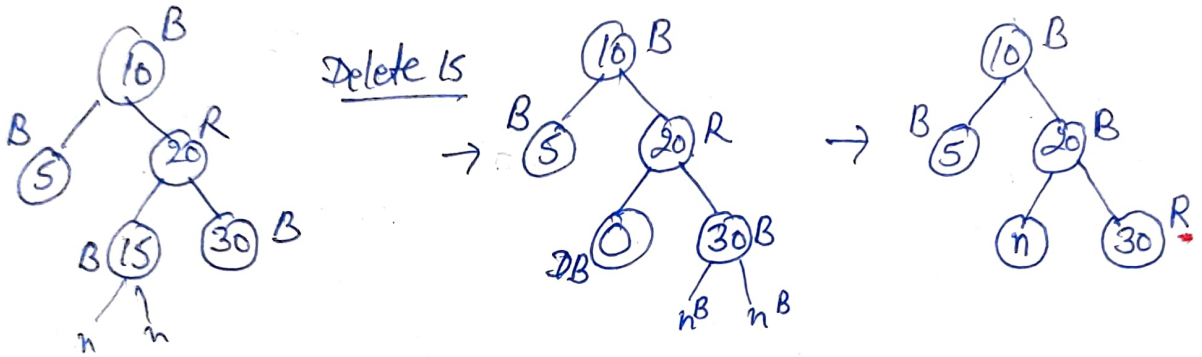


~~Case 2~~ If Root is ~~black~~ Double black, Just remove DB

Case 3 If DB's sibling is black and both its childrens are black

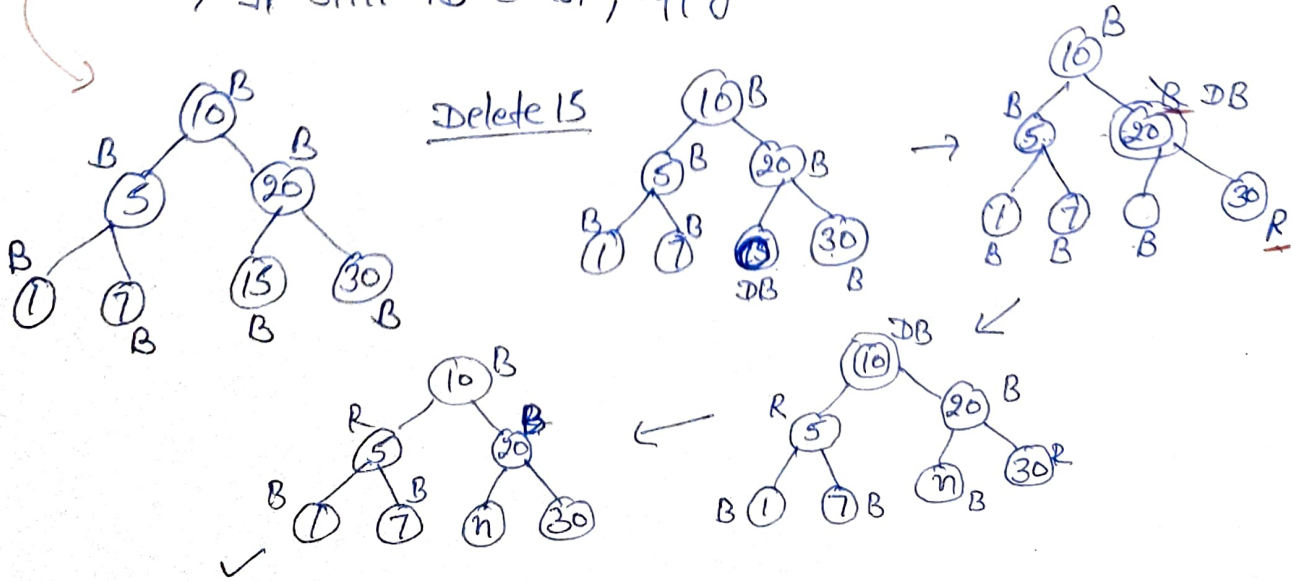
↳ Remove DB
↳ Add Black to Parent ① If P is red





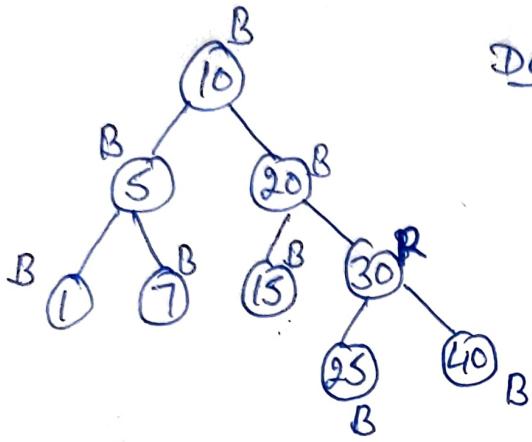
Case 3 If DB's sibling is black and both its childrens are black

— Remove DB
→ Add black to Parent P
↳ If P is red, it becomes black
↳ If P is black, it becomes double black
→ Make Sibling red
→ If still DB exist, apply other case.

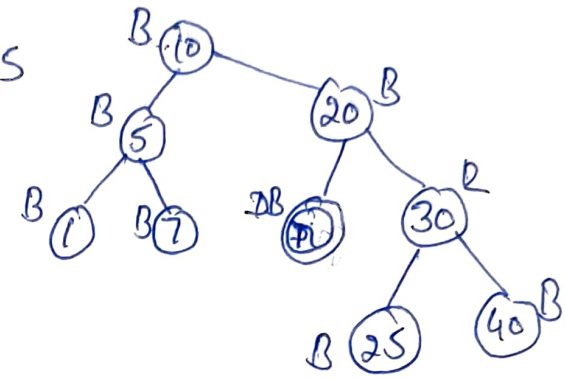


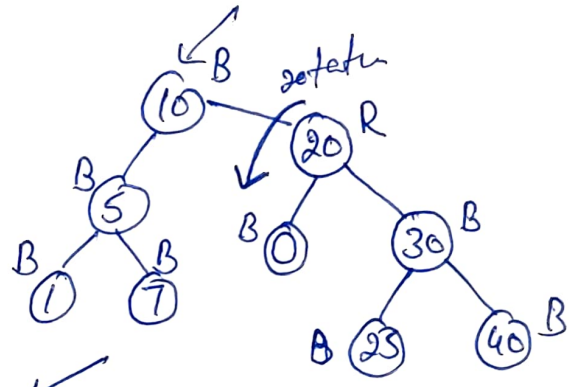
Case 4 If DB's sibling is red.

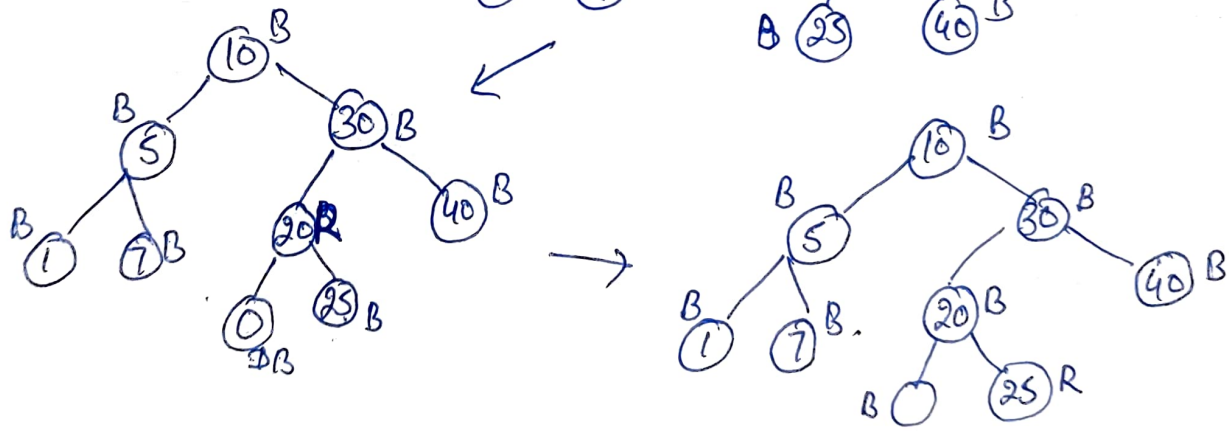
→ Swap color of Parent and its sibling.
→ rotate the Parent in DB direction.
→ reapply cases.



Delete 15

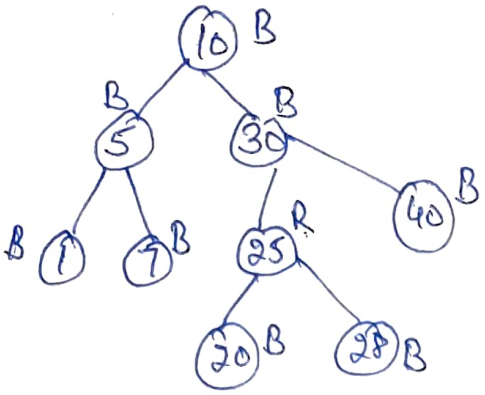




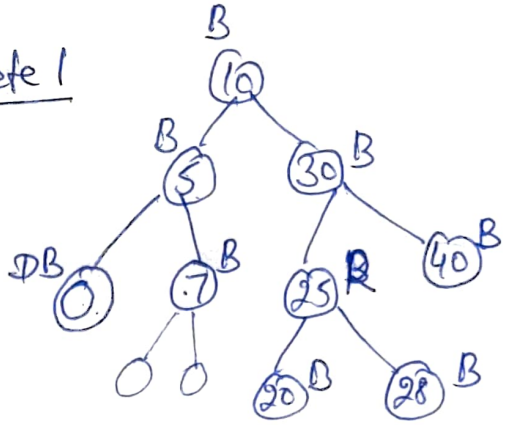


Case 5 DB's sibling is black, sibling child who is far from DB is black but near child to DB is red.

→ swap colors of DB's siblings and siblings's children who is near to DB
→ rotate sibling in the opposite direction of DB
→ apply case 6

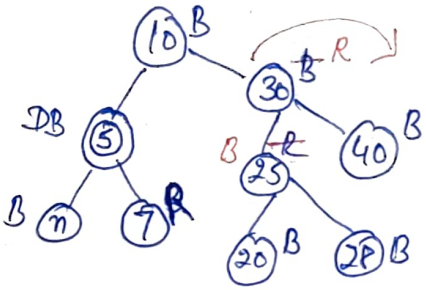


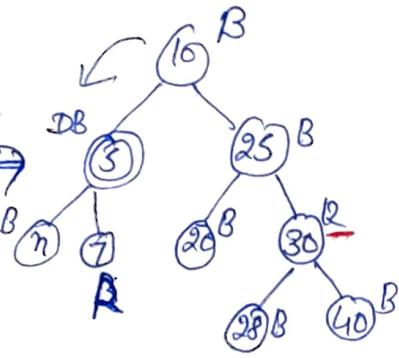
Delete 1



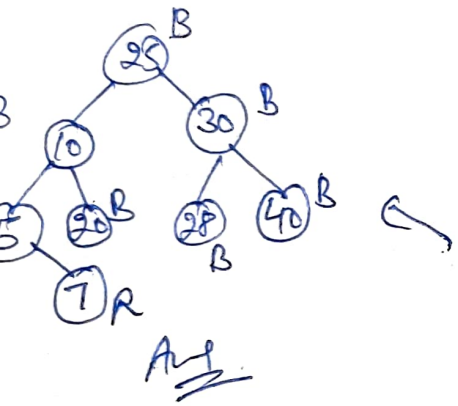
3-5-6

Case 3



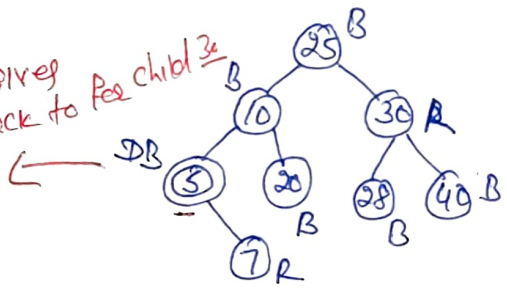


Case 6
Rotate 10 towards 5



Ans

(5) now gives extra black to far child

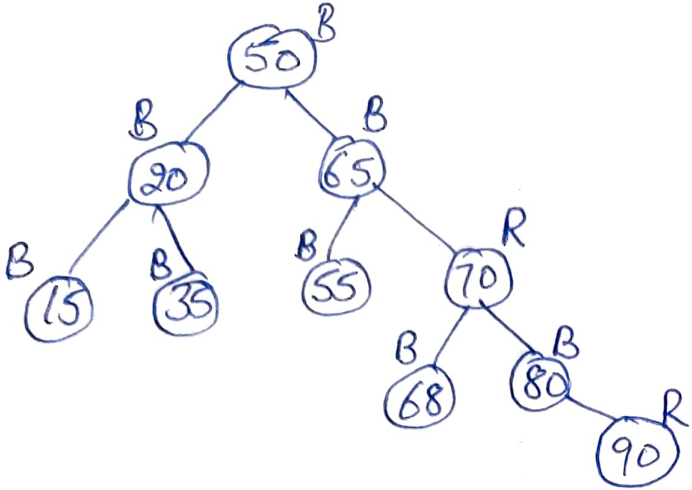


Case 6 DB's sibling is black far child is red

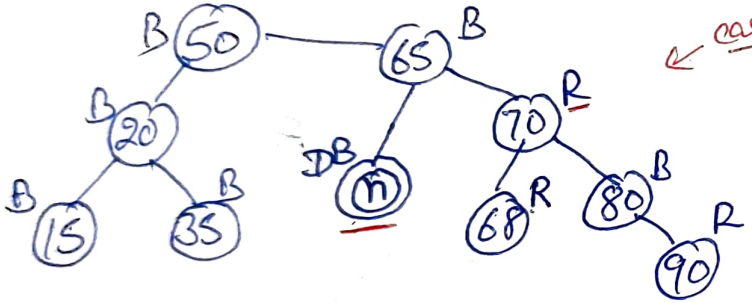
→ swap color of Parent and sibling
→ rotate Parent in DB's direction.
→ remove DB
→ change color of red child to black

Perform Deletion in RB-Tree →

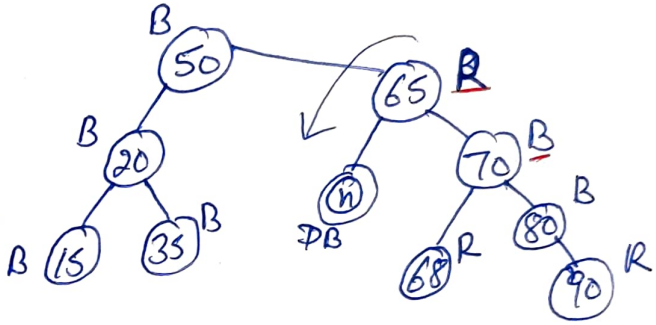
55, 30, 90, 80, 50, 35, 15



① Delete 55

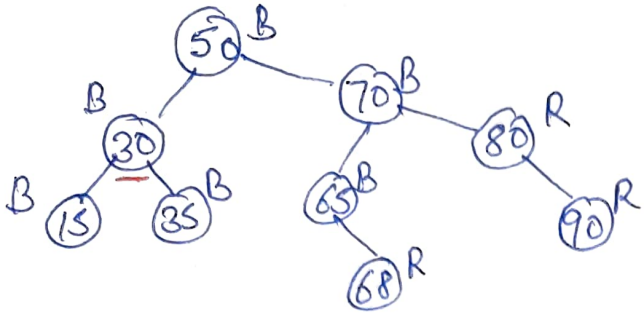


case 4 →

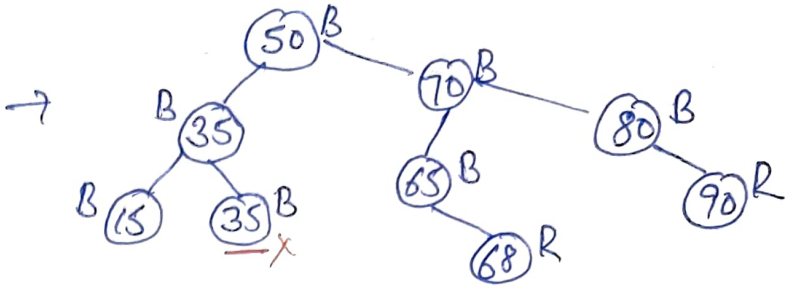


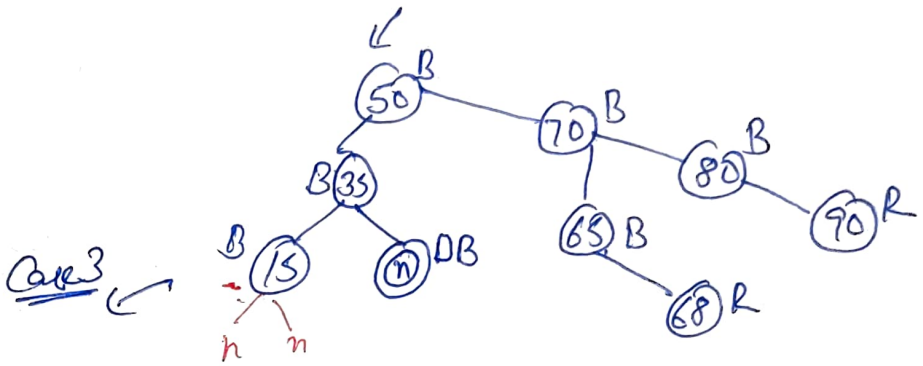
↓

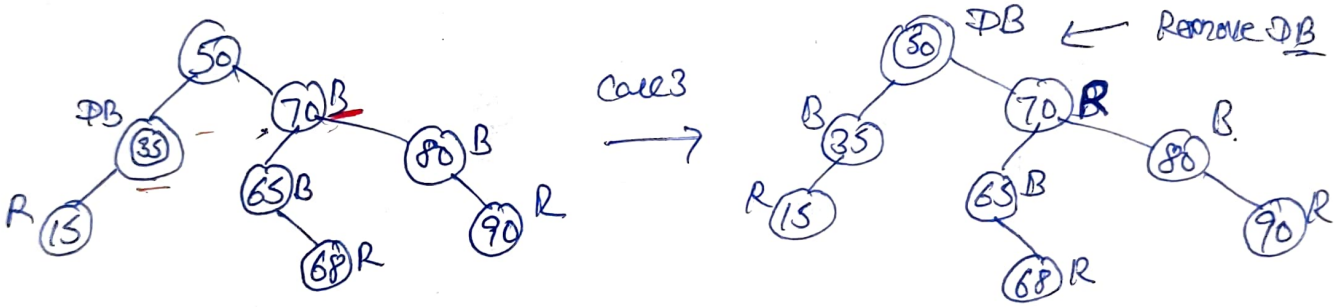
Delete 30



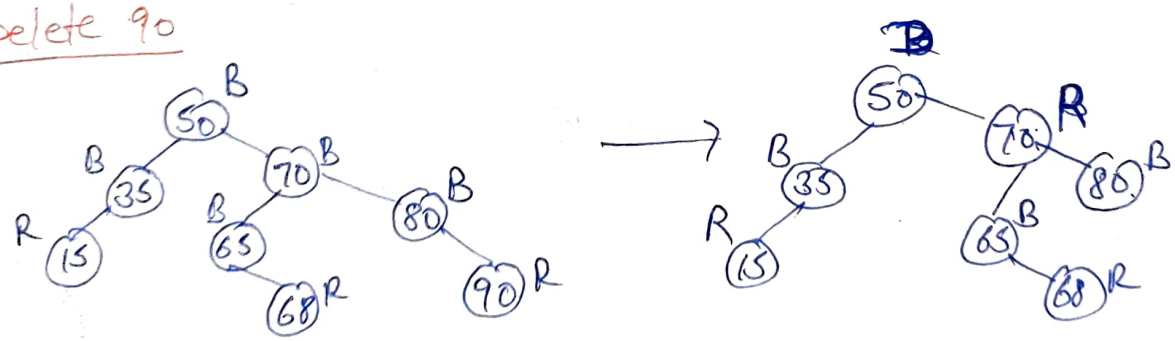
Replace 30 with its inorder successor 35



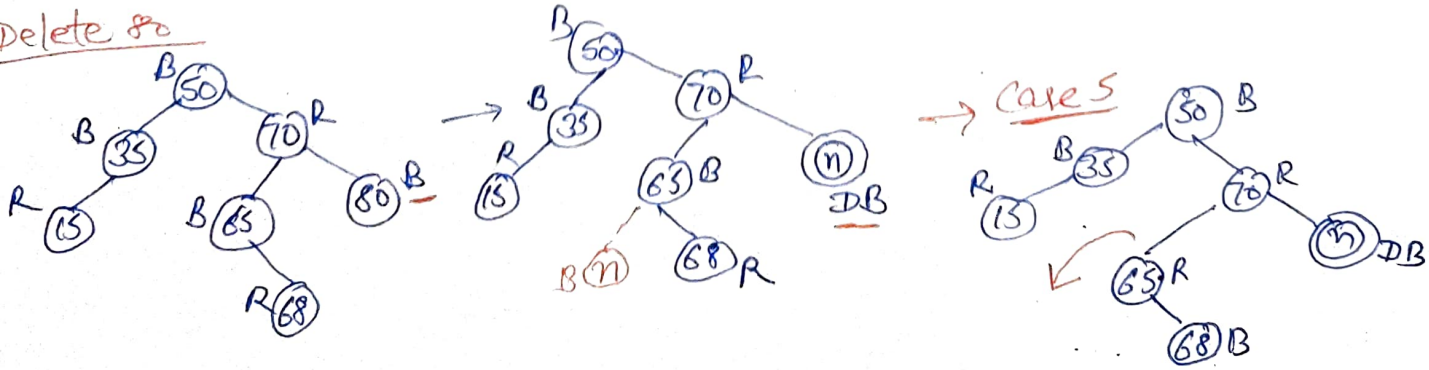


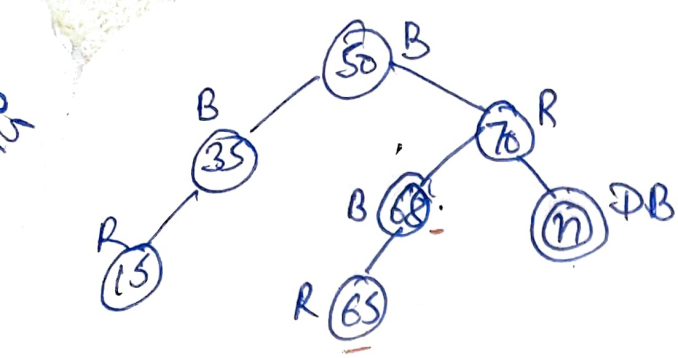


Delete 90

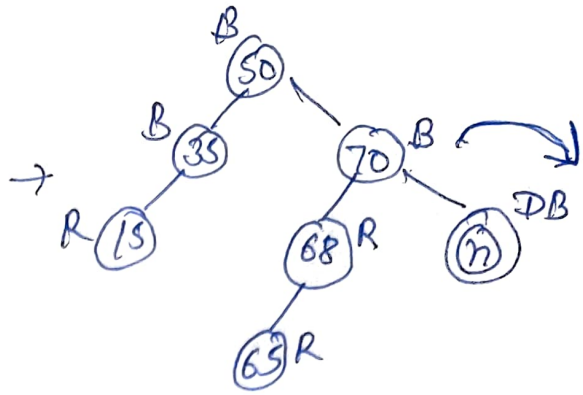


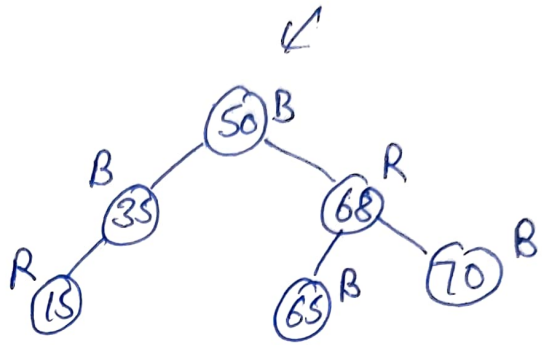
Delete 80



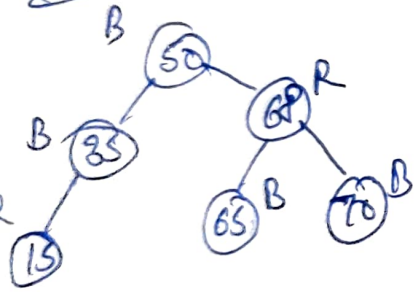


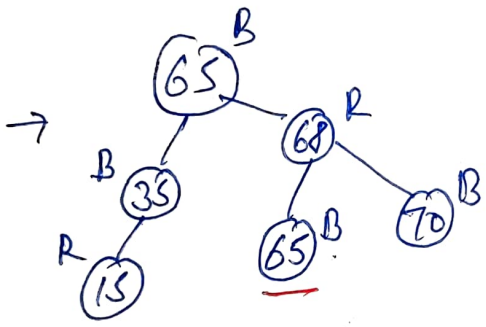
case 6

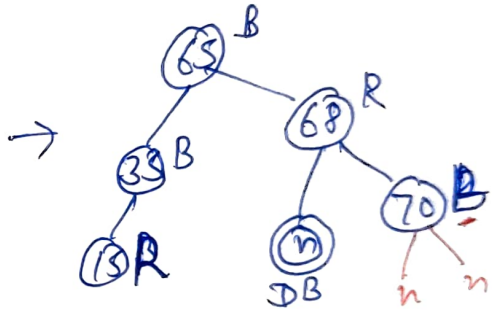




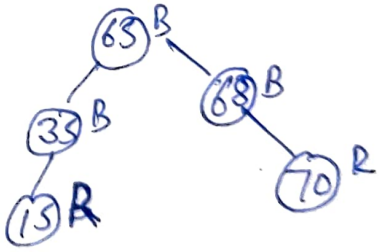
Delete 50



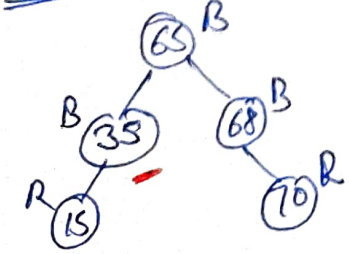


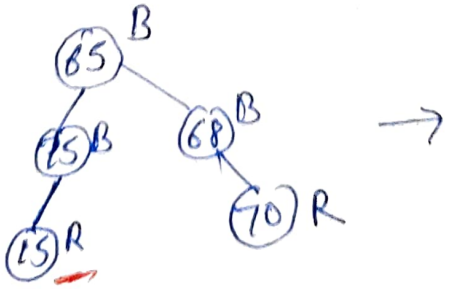


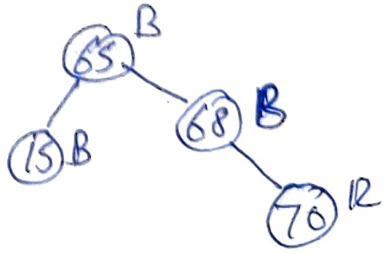
case 3



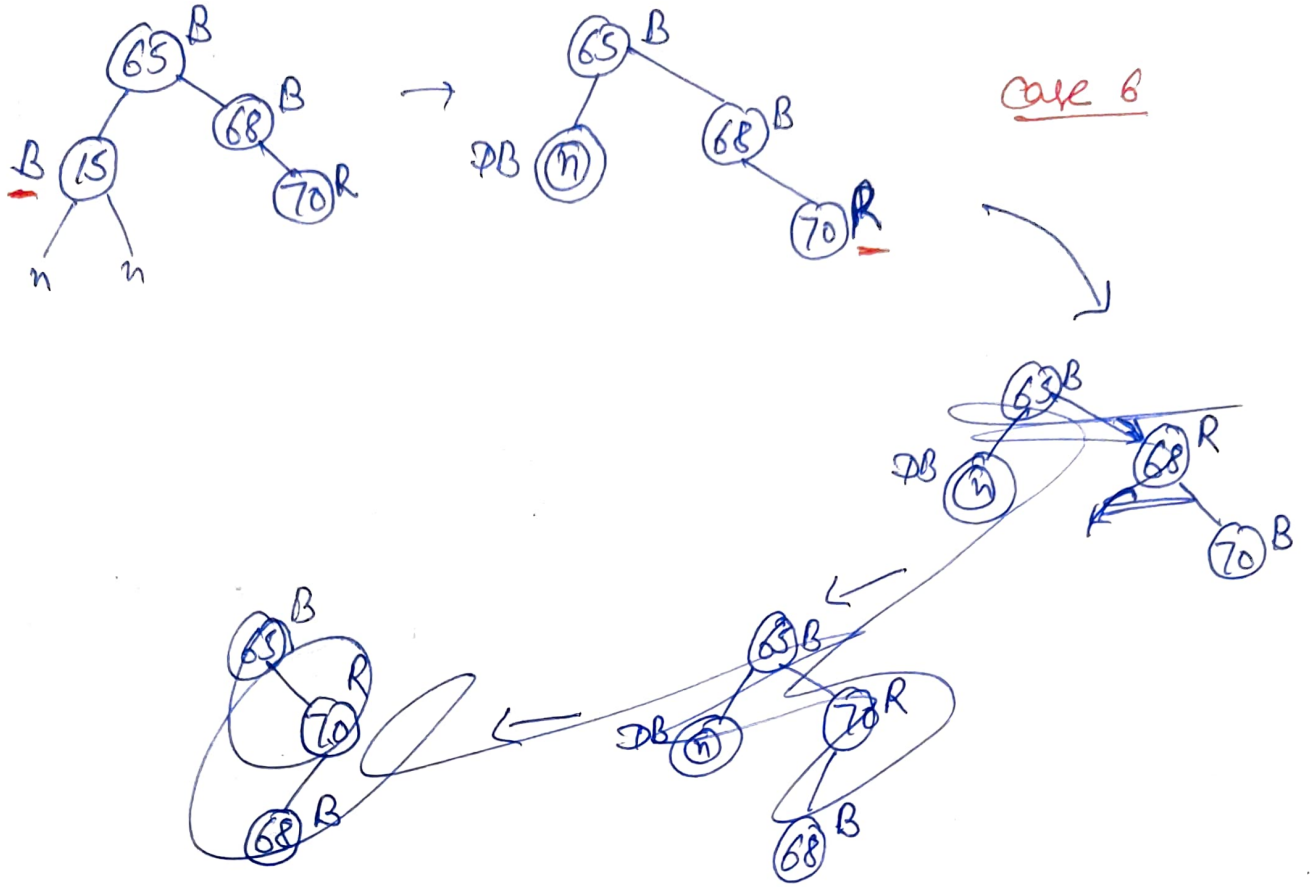
Delete 35

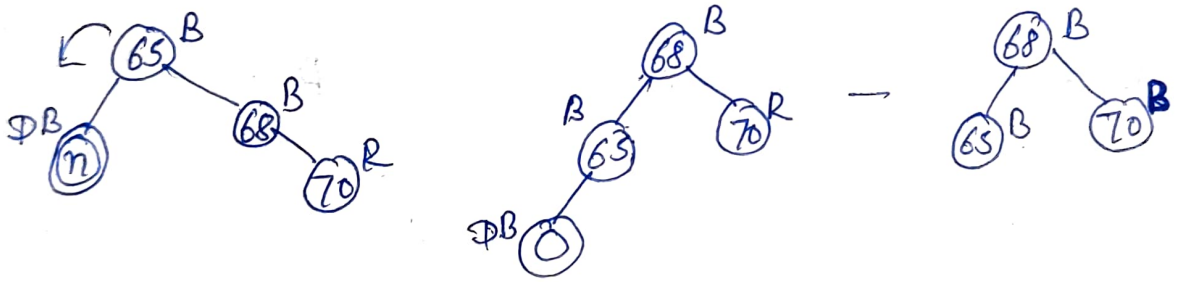




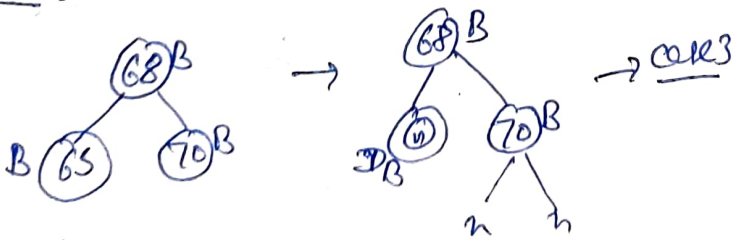


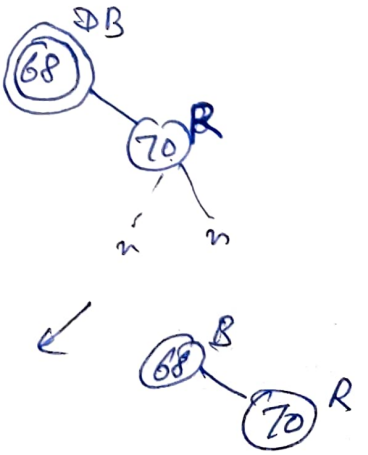
Delete 15





Delete 65



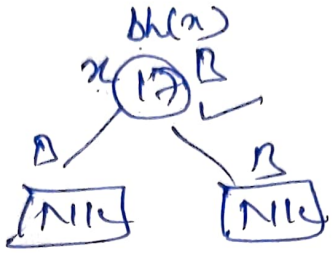


A-RB tree with n internal nodes has height at most $2\lg(n+1)$

the subtree rooted at any node x contain atleast

$2^{bh(x)} - 1$ internal node.

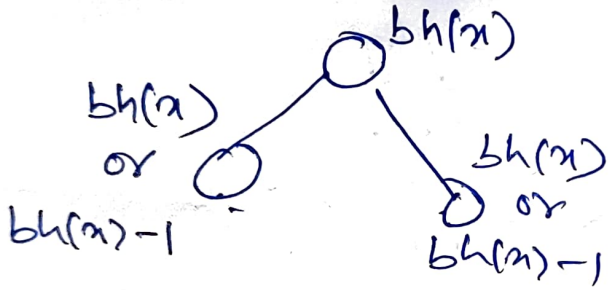
let $x = 17 \Rightarrow$



$2^1 - 1 = 1$

$x = 41 \Rightarrow bh(x) = 2$

$2^2 - 1 = 3$ Internal node



$h \leq 2bh(x)$

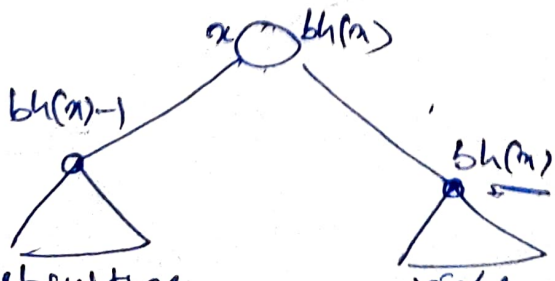
$\Rightarrow h = 2bh(x)$

$\Rightarrow \boxed{bh(x) = h/2}$

$bh(x) \Rightarrow$ atleast $2^{bh(x)} - 1$ internal node

$bh(x)-1 \Rightarrow$ atleast $2^{bh(x)-1} - 1$ internal node

} minimum



atleast $2^{bh(x)-1} - 1$ internal node

atleast $2^{bh(x)-1} - 1$ internal node

$n \geq (2^{bh(x)-1} - 1) + (2^{bh(x)-1} - 1) + 1$

$n \geq 2^{bh(x)} - 1$

$n+1 \geq 2^{bh(x)}$

$n+1 \geq 2^{h/2}$

$$n+1 \geq 2^{h/2}$$

$$\Rightarrow 2^{h/2} \leq n+1$$

$$\Rightarrow \log_2(2^{h/2}) \leq \log_2(n+1)$$

$$\Rightarrow h/2 \leq \log_2(n+1)$$

$$\Rightarrow \boxed{h \leq 2\log_2(n+1)}$$

$H_1 = 7$ + = 28

$H_2 = 18$