Smita Attarde

Module 3

Red Black Trees

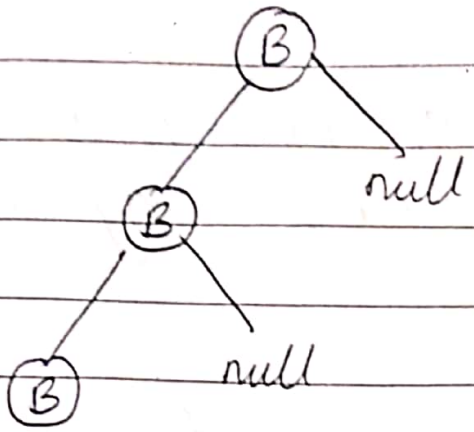
→ It is a self-balancing binary search tree (BST)
→ Each node in the red black tree is colored with either red or black color.
→ The color of each node is decided based on the properties of the RBT.

* properties of RBT→
1) Every node is colored either red or black.
2) The root is always black.
3) Every null pointer is black.
4) If a node is red, then both its children are black.
5) New node added is always red.
6) Every single path from a node to a descendant leaf contains the same number of black nodes.

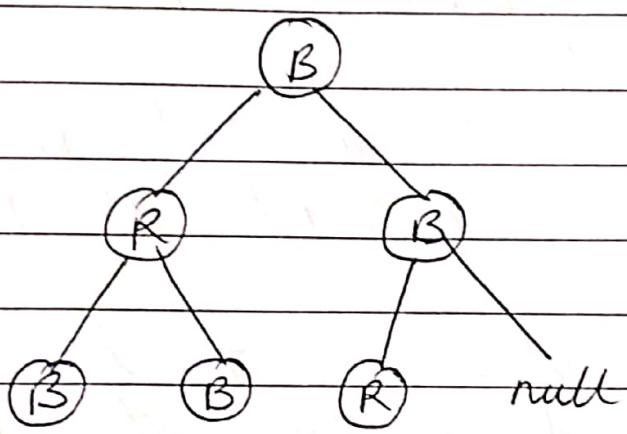
In short,
Rules:-
1) Root is black.
2) No red-red parent-child relationship.
3) Number of black nodes from root to leaf node is same.

Smita Attarde

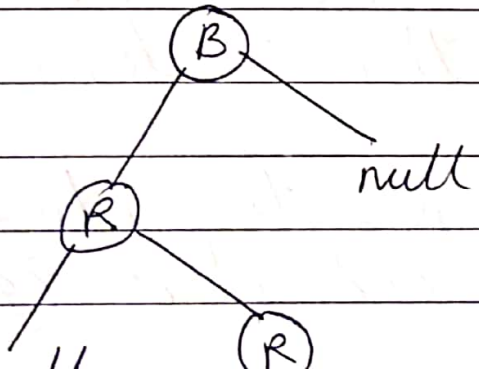
classmate
Date
Page 109



Not an RBT



This is an RBT



Not an RBT

Smita Attarde

* RBT Insertion →

→ Insertion of new node in RBT is done in the same way as it is done in BST.

Algorithm to insert a node in RBT →

1) If the tree is empty, then create the black root node.
2) Colour the new node as red and insert it.
3) If the new node's parent is black, then done.
4) If the new node's parent is red, then then exists red-red parent child relationship.
   a) If the new node's parent's sibling is black or absent, then rotate, recolour and done.
   b) If the new node's parents sibling is red, then recolour and check again.

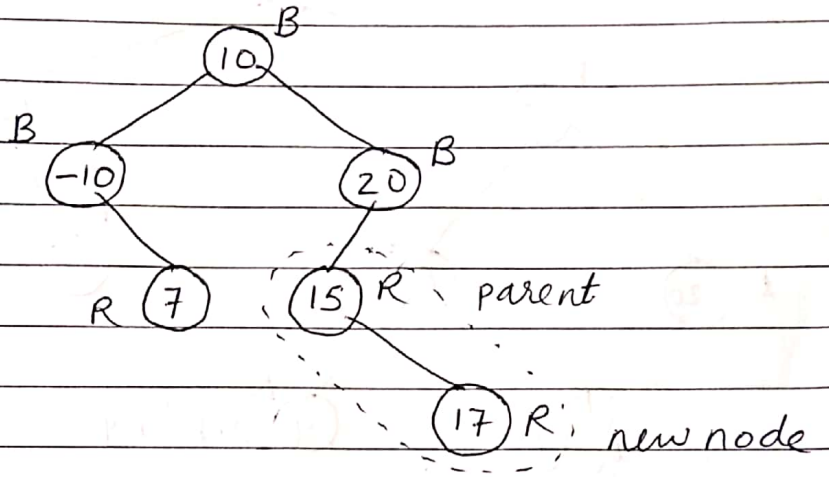
Note:
Here,

→ Recolour means make the parent and uncle as black and their parent as red
→ Rotations are same as AVL tree rotati

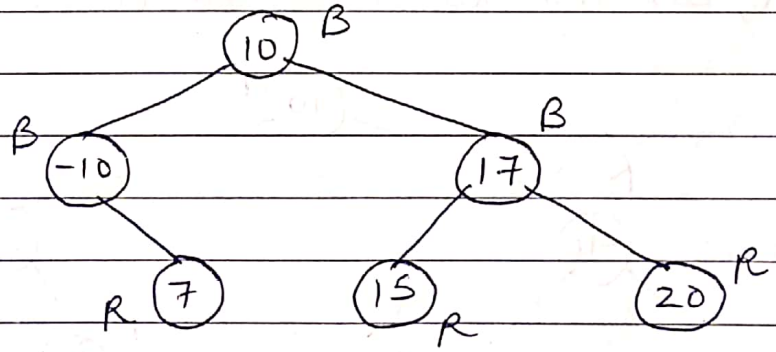
~~eg if sibling is mis~~

eg if new node's parent's sibling is missing,

Consider the RBT, in which 17 is inserted as new node.



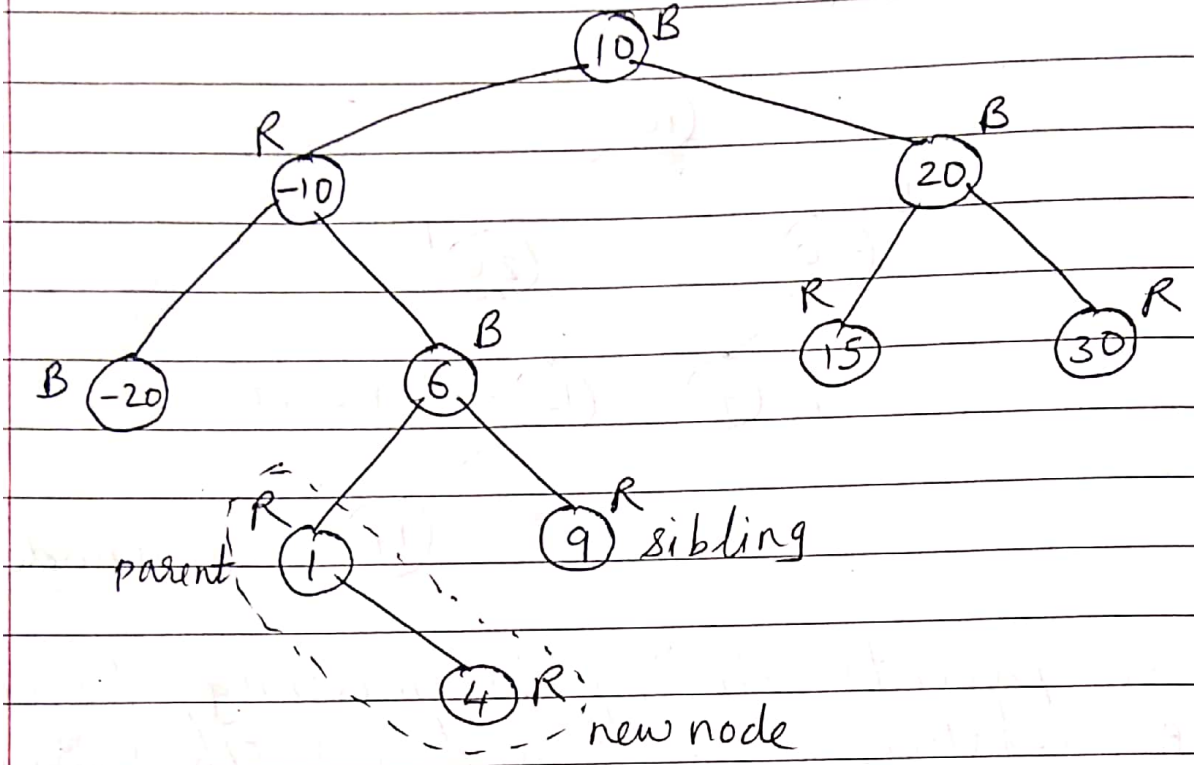
Here, parent's sibling is missing hence LR rotation is performed an then recolouring is done accordingly.



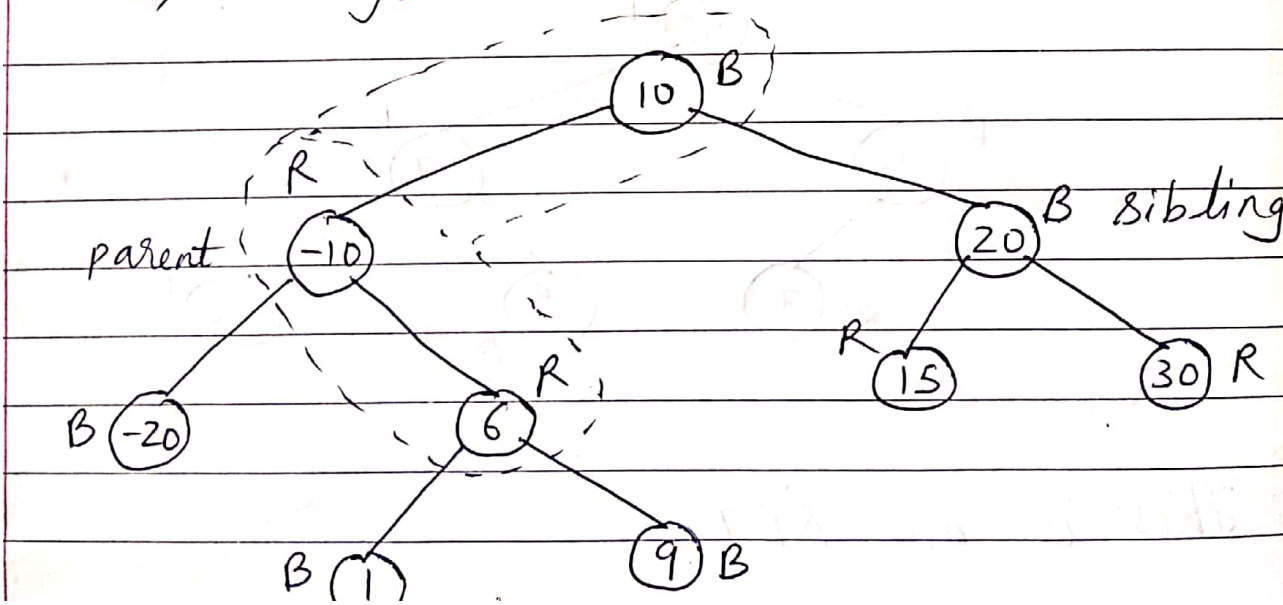
This is an RBT.

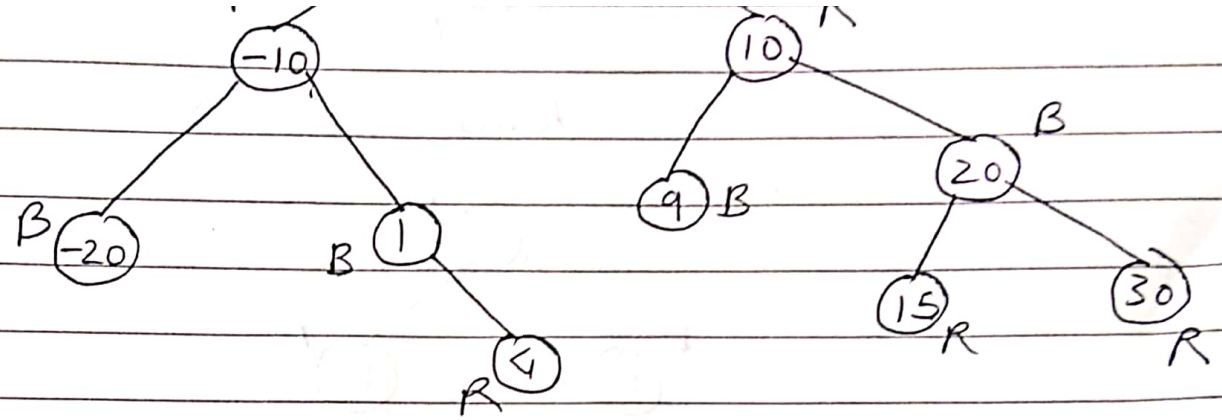
if ~~parent's~~ new node's parent's sibling is red

Consider the RBT, in which 4 is inserted as new node.



Here, sibling is red, hence recolour.





This is an RBT.

Solved Examples:-

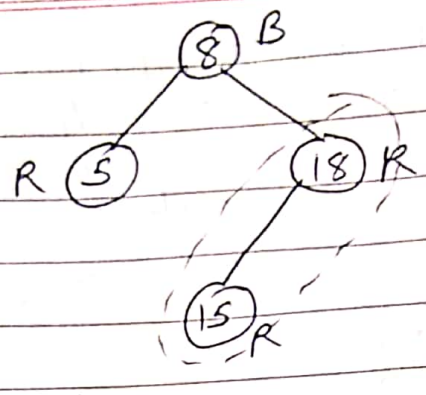
1) Insert the following elements in a RBT
8, 18, 5, 15, 17, 25, 40 and 80

① 8 → 8 B

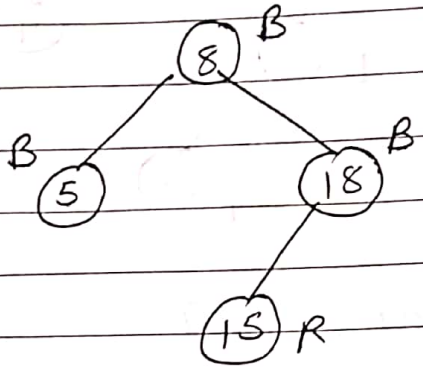
② 18 → 8 B, 18 R

③ 5 → 8 B

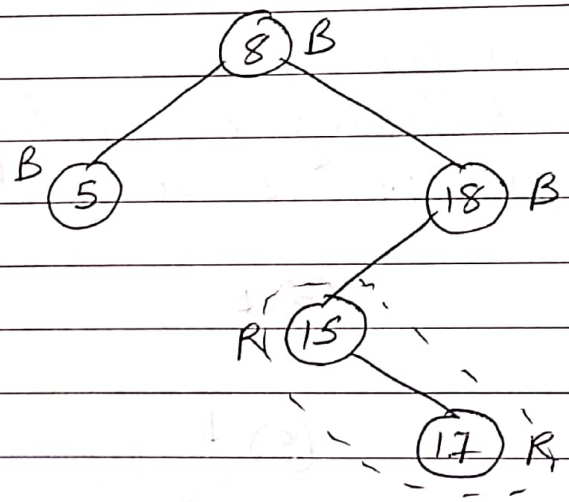
(4) 15 →



sibling is red, hence recolor

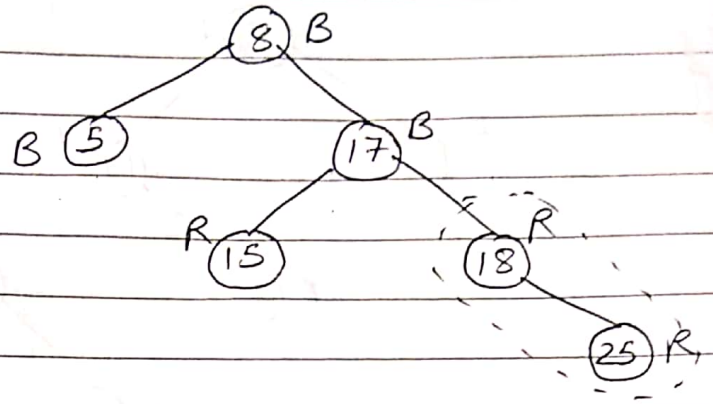


(5) 17 →

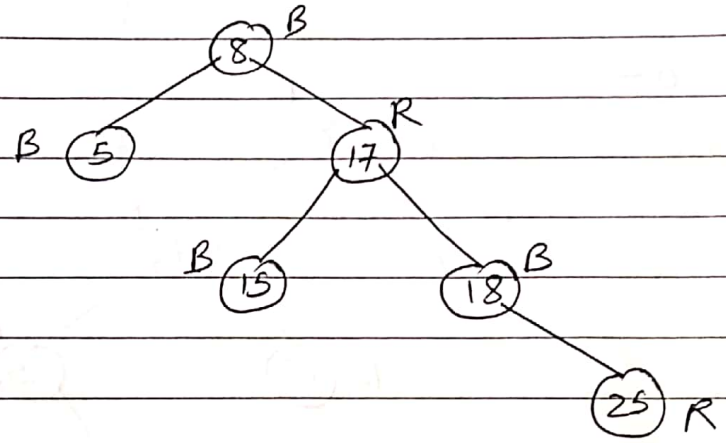


Smita Attarde

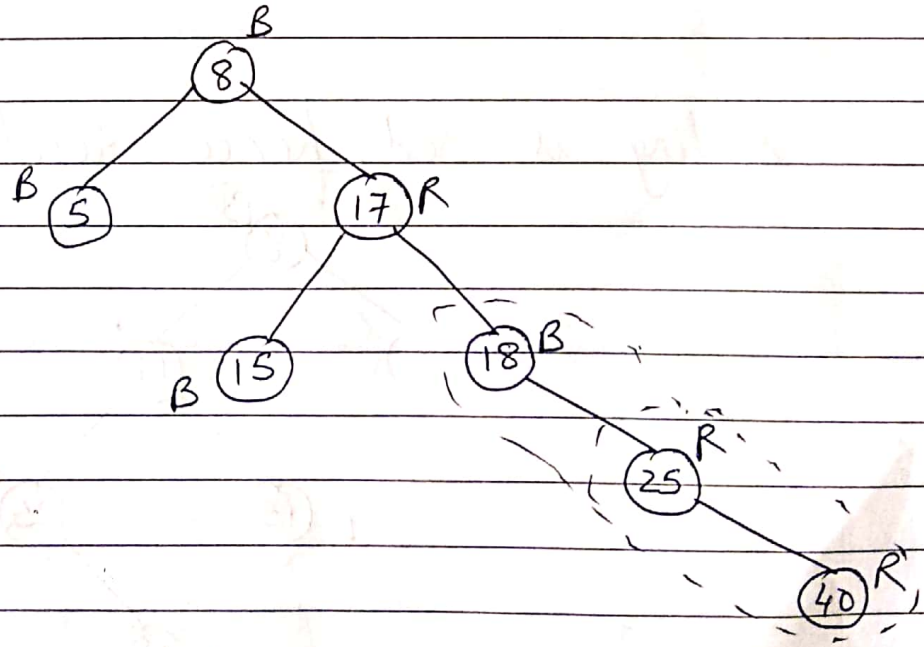
(6) 25 →



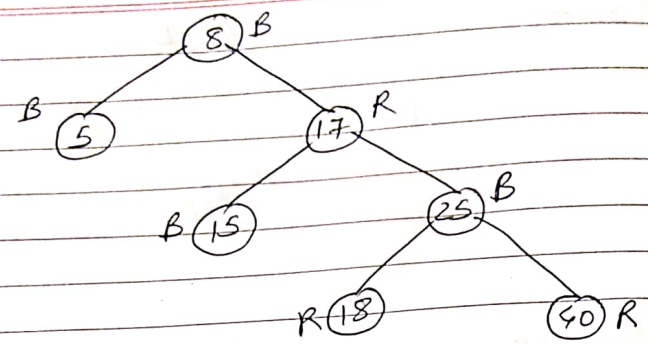
sibling is red, hence recolour.



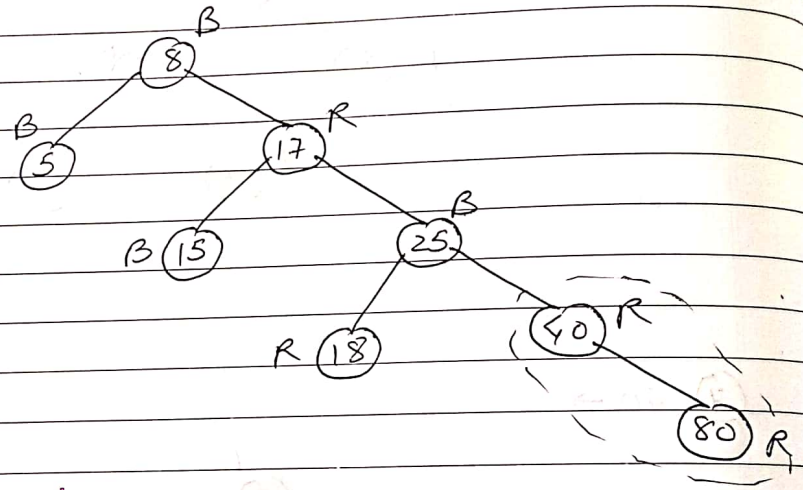
(7) 40 →



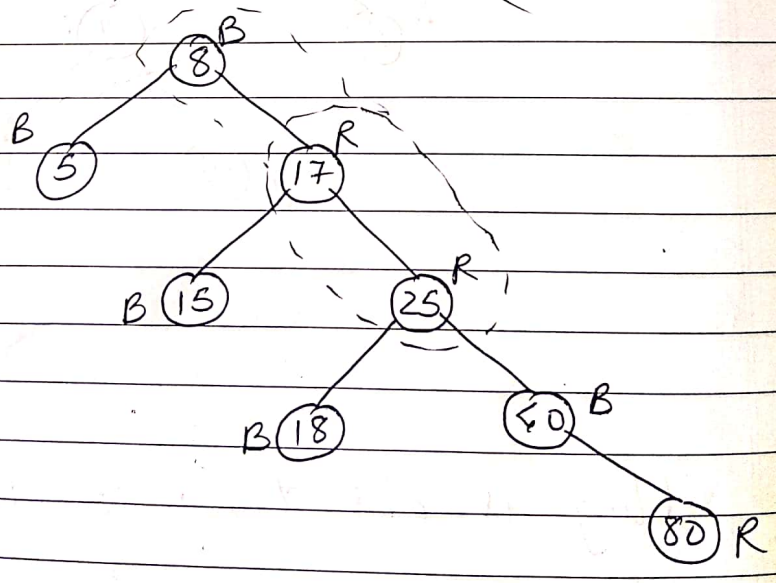
sibling is absent, hence RR rotation



(5) 80 →



sibling is red, hence recolour



Smita Attarde

sibling is

this

2) Create

2,

(1) 2 →

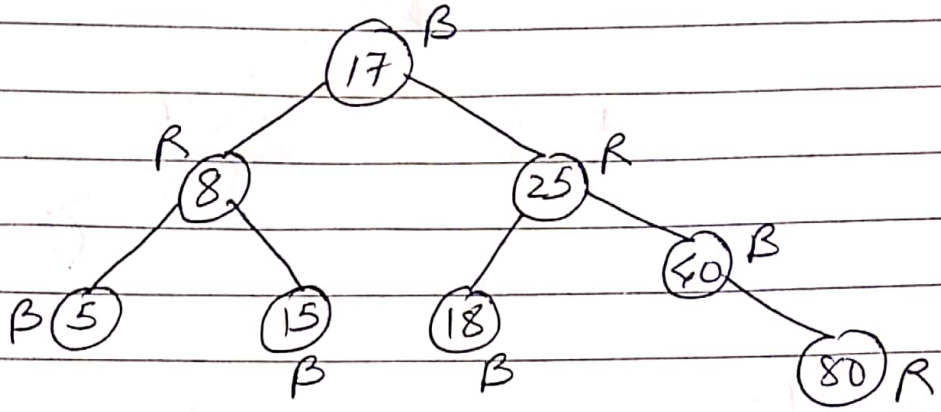
(2) 1 →

(3) 4 →

(4) 5 →

sib

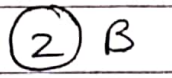
sibling is black, hence RR rotation



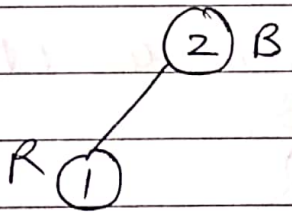
this is final RBT.

2) Create RBT for the following elements:-
2, 1, 4, 5, 9, 3, 6, 7

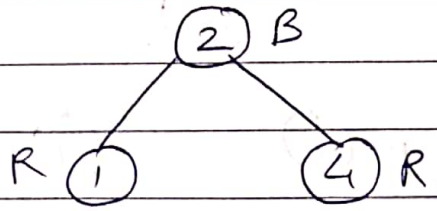
(1) 2 →



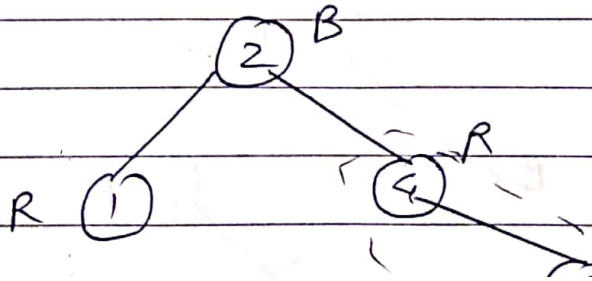
(2) 1 →

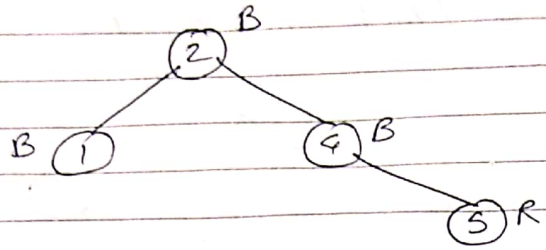


(3) 4 →

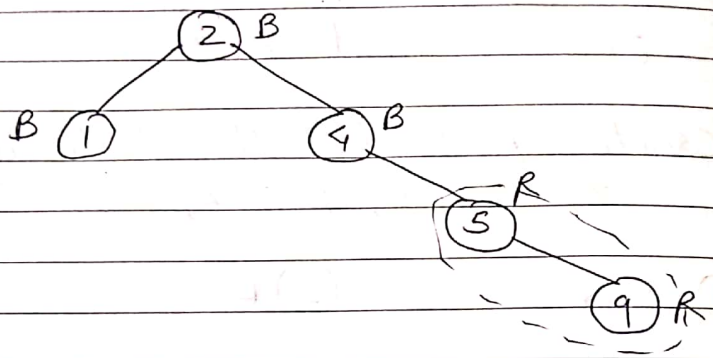


(4) 5 →

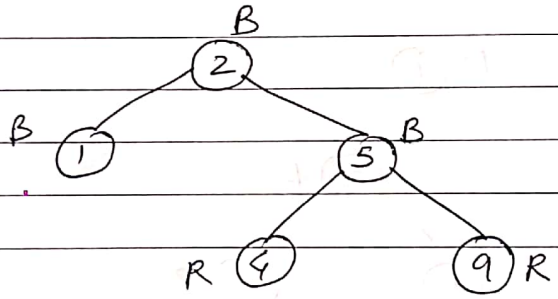




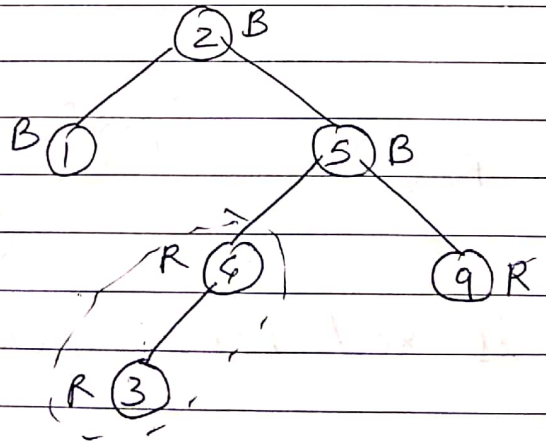
(5) 9 →



sibling is absent, hence RR rotation



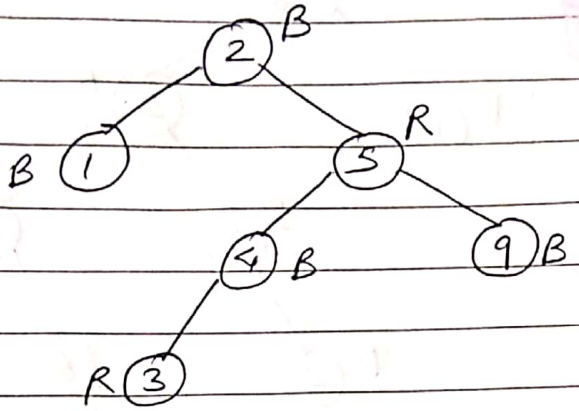
(6) 3 →



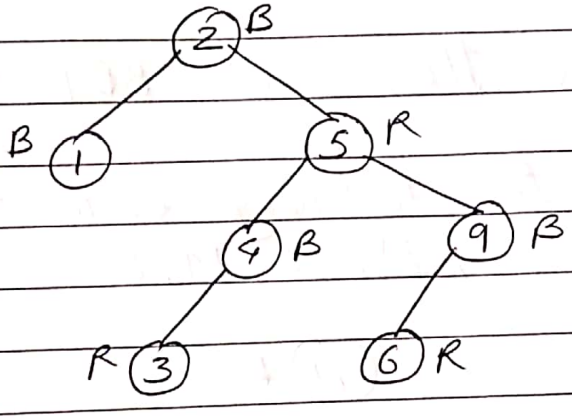
Smita Attarde

Smita Attar

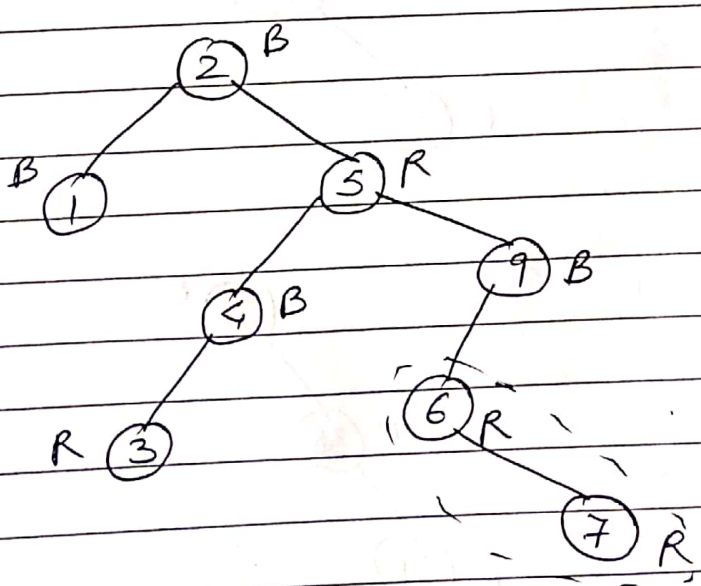
sibling is red, hence recolour



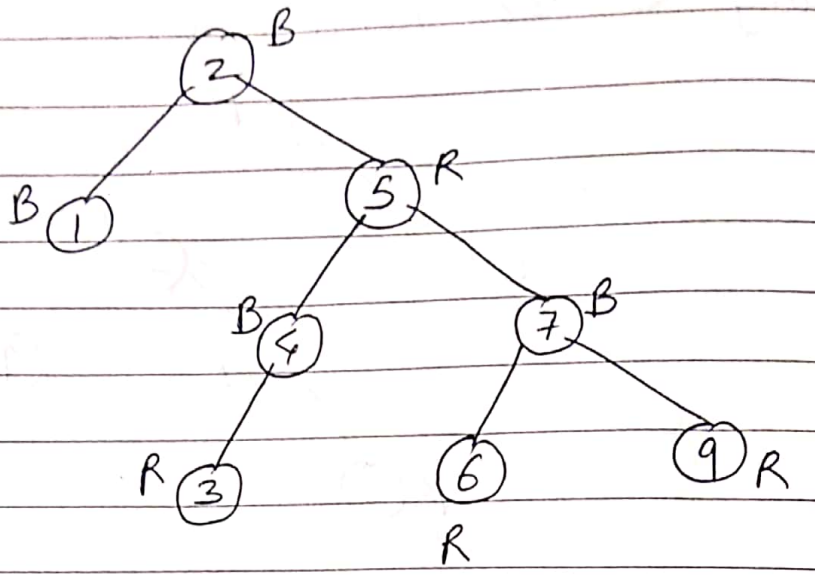
(7) 6 →



(8) 7 →



sibling is absent, hence LR rotation.

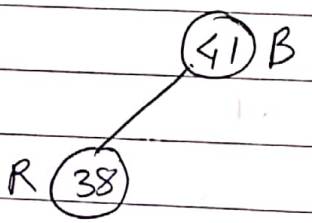


This is the final RBT

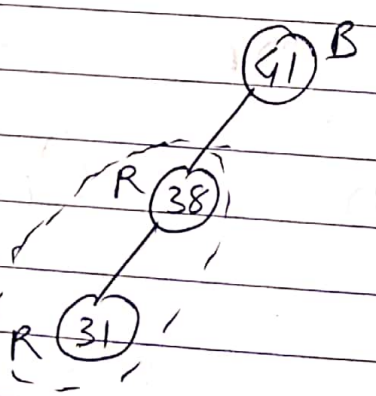
3) Insert the following element in RBT
41, 38, 31, 12, 19, 8.

(1) 41 → 41 B

(2) 38 →

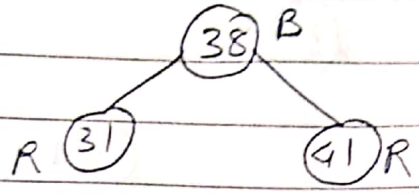


(3) 31

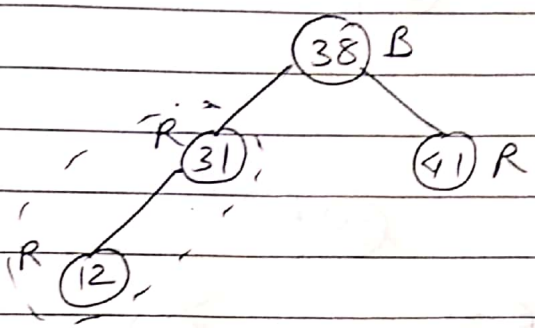


Sibling

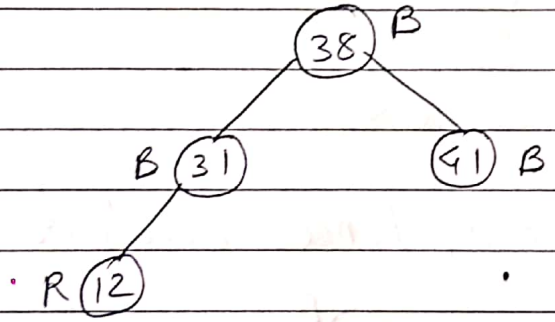
Smita Attarde



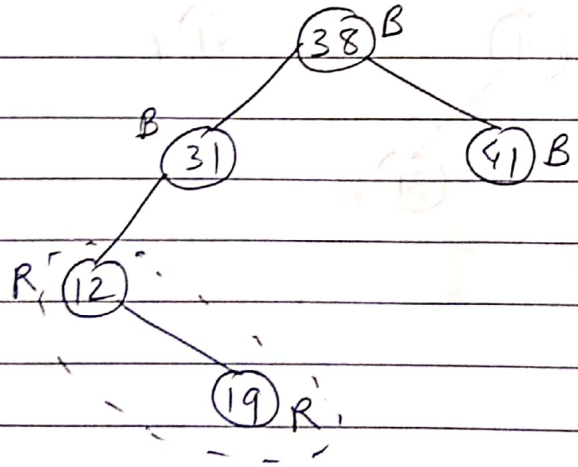
(4) 12→



sibling is red, hence recolour



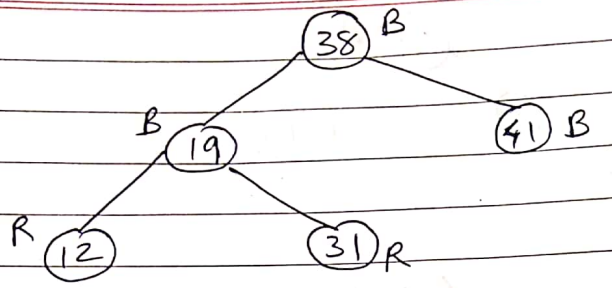
(5) 19→



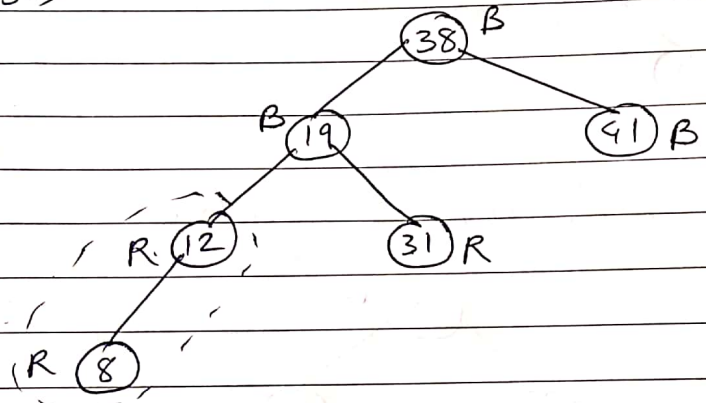
sibling is absent, hence LR rotation.

Smita Alvarae

classmate
Date
Page 122



6. 8→

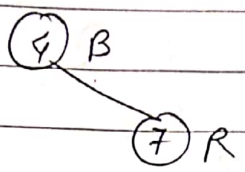


sibling is red hence recolour

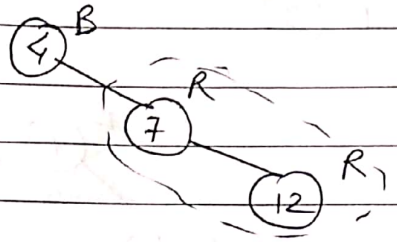
Create RBT for the following elements
4, 7, 12, 15, 3, 5, 14, 18, 16, 17

① 4 →

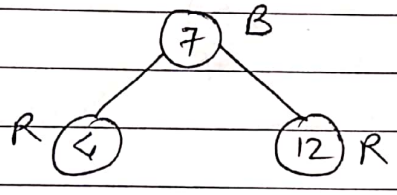
② 7 →



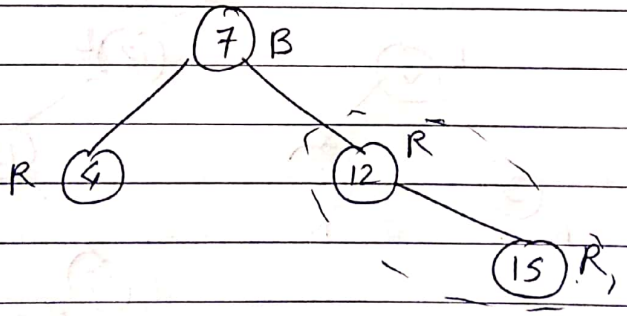
③ 12 →



sibling is absent, hence RR rotation



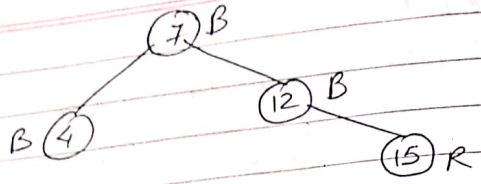
④ 15 →



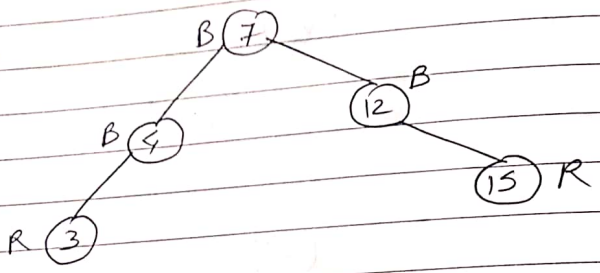
sibling is red, hence recolour.

Smita Attarde

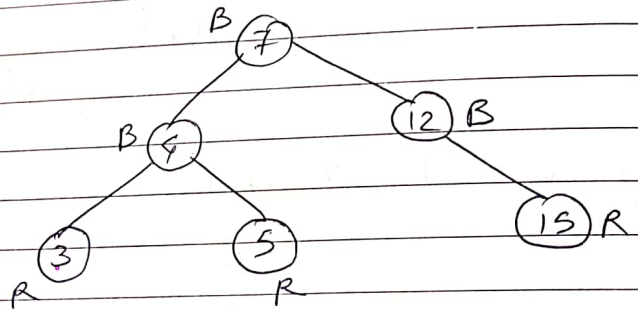
Date
Page 124



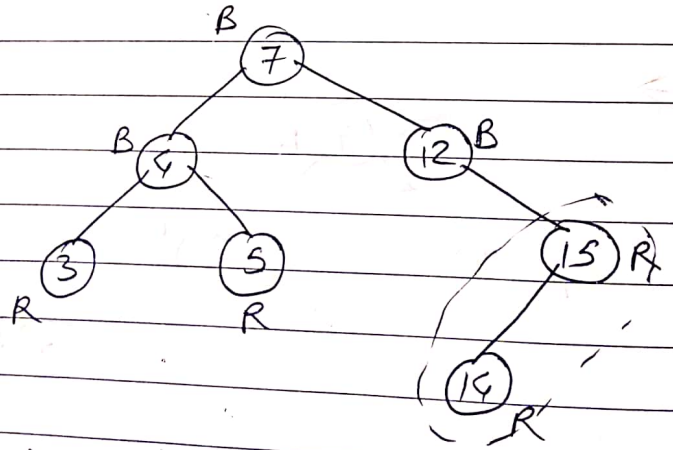
(5) 3 →



(6) 5 →



(7) 14 →



sibling is absent, hence RL rotation

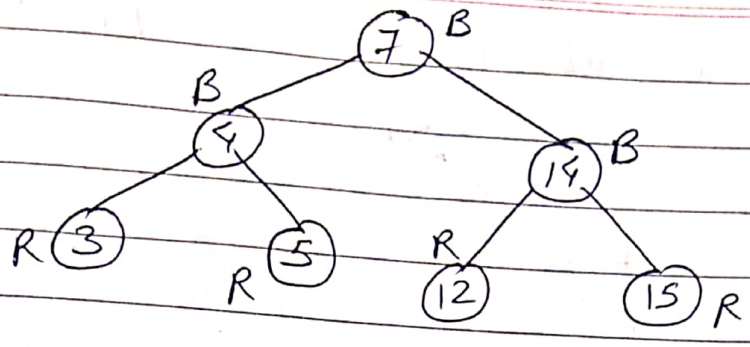
(8) 18 →

sibling

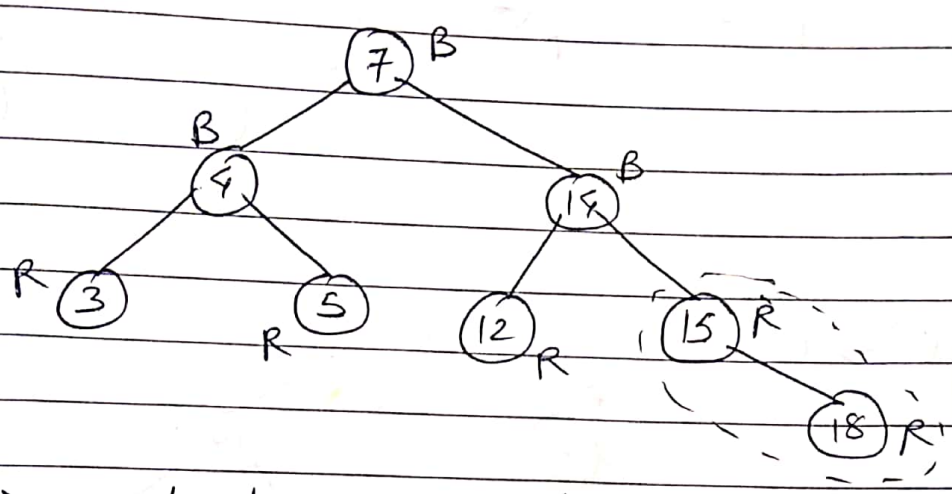
(9) 16 -

Smita Attarde

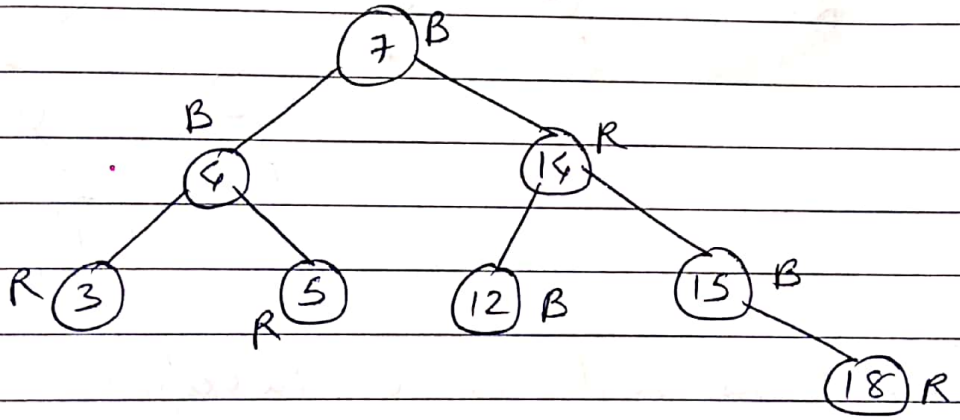
classmate
Date
Page 125



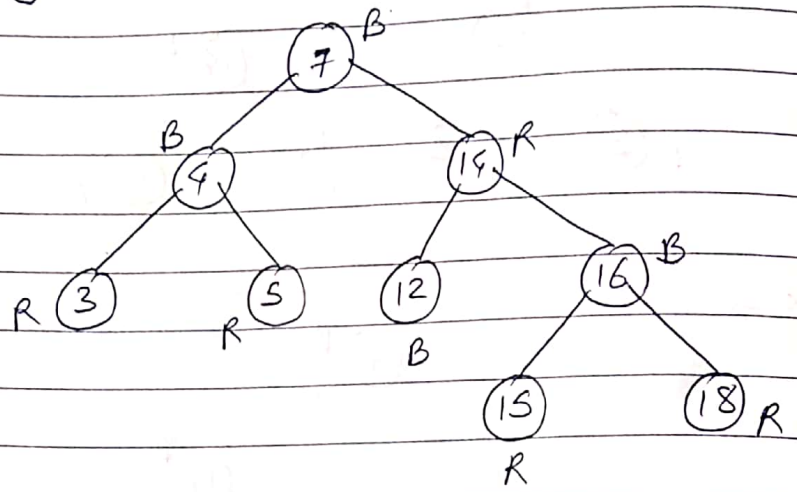
(8) 18 →



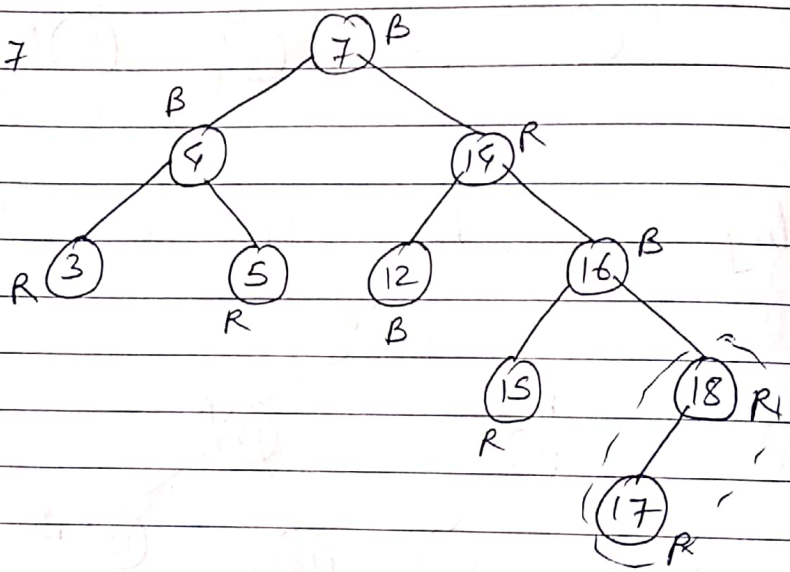
sibling is red, hence recolour



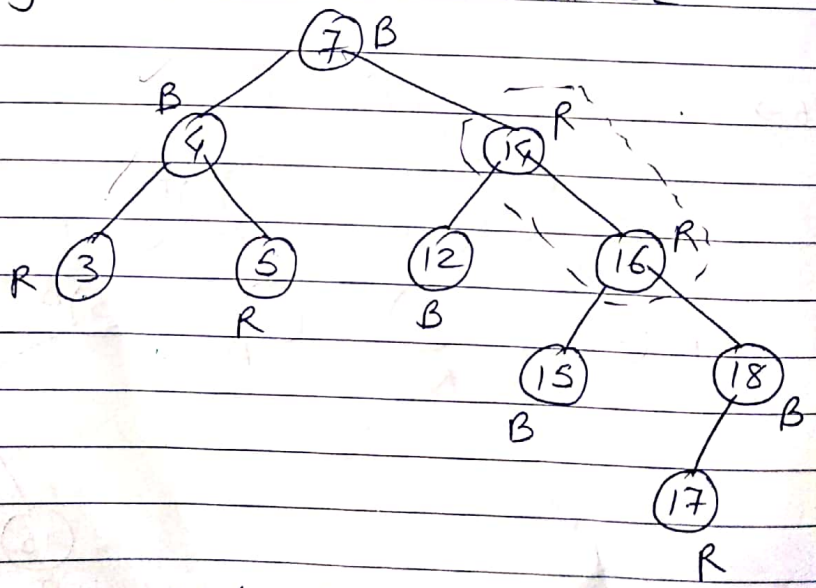
sibling is absent; hence RL rotation



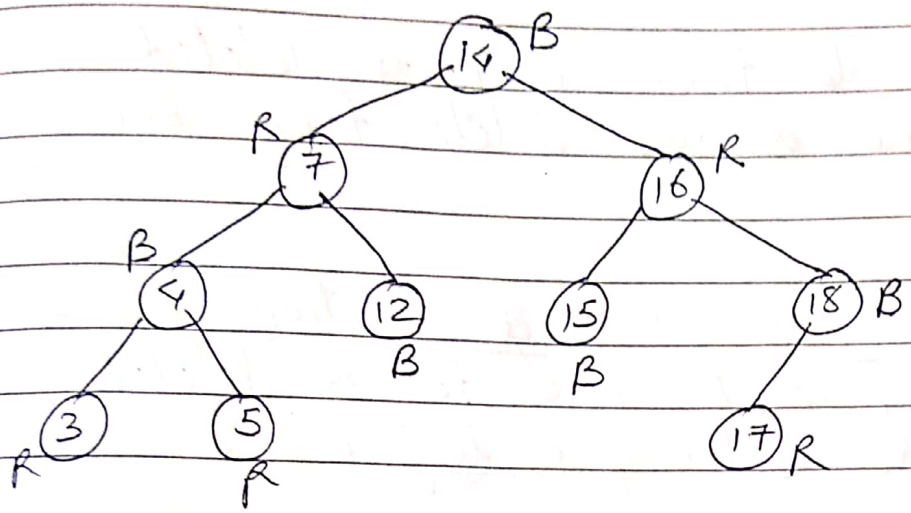
10 → 17



sibling is red, hence recolour



sibling is black, hence RR rotation



This is the final RBT

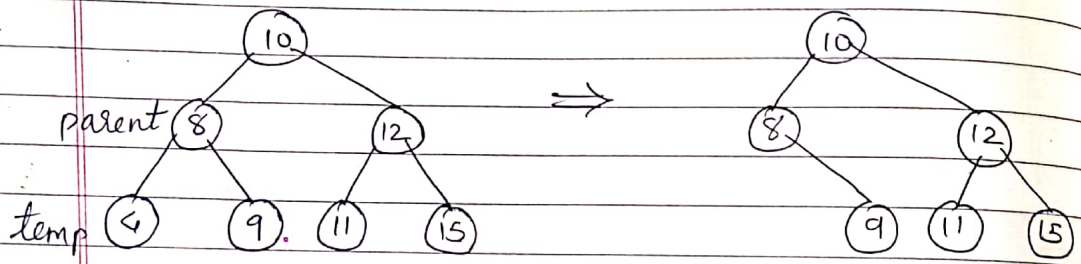
Red Black Tree Deletion →

→ A node from RBT is deleted in similar manner as we delete from binary search tree.

* Binary Search Tree Deletion cases →
temp is the node to be deleted.

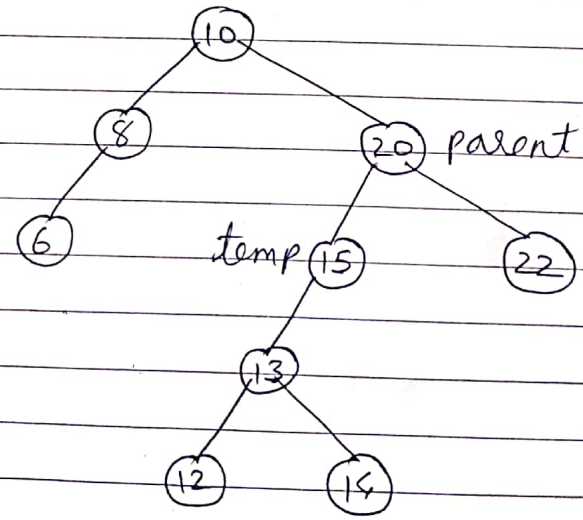
Case 1:- Deleting a leaf node.

eg)

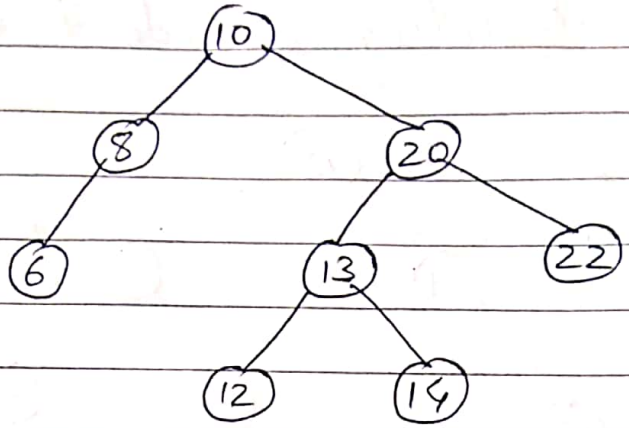


Case 2:- Deleting a node having only left child.

eg)

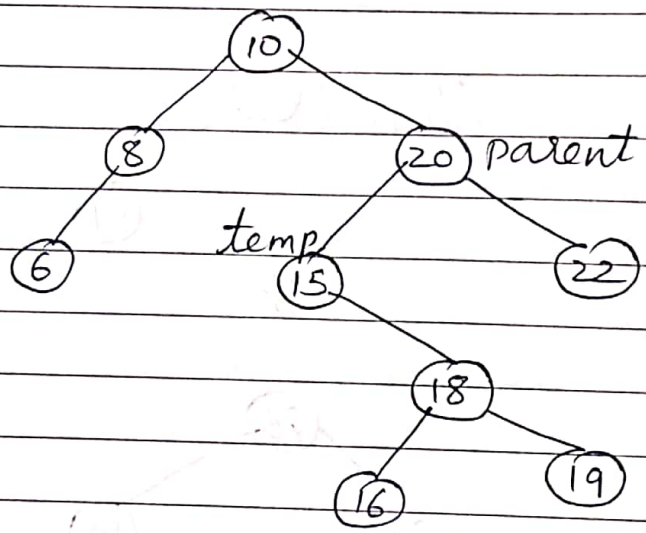


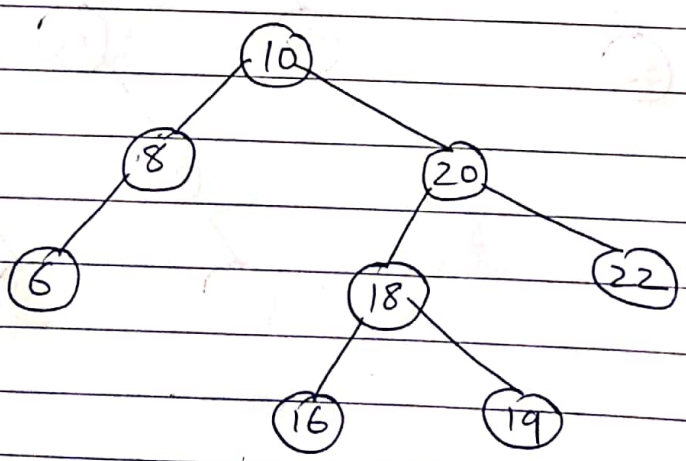
⇒



Case 3:- Deleting a node having only right child.

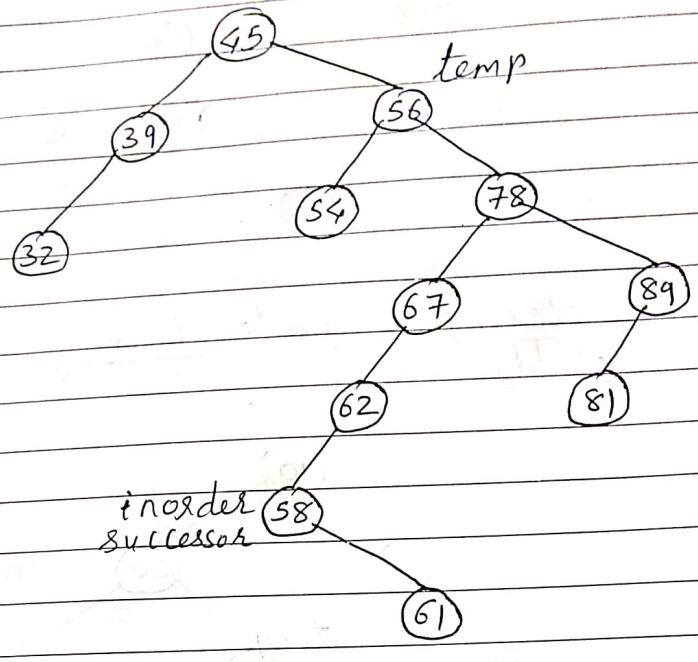
eg)

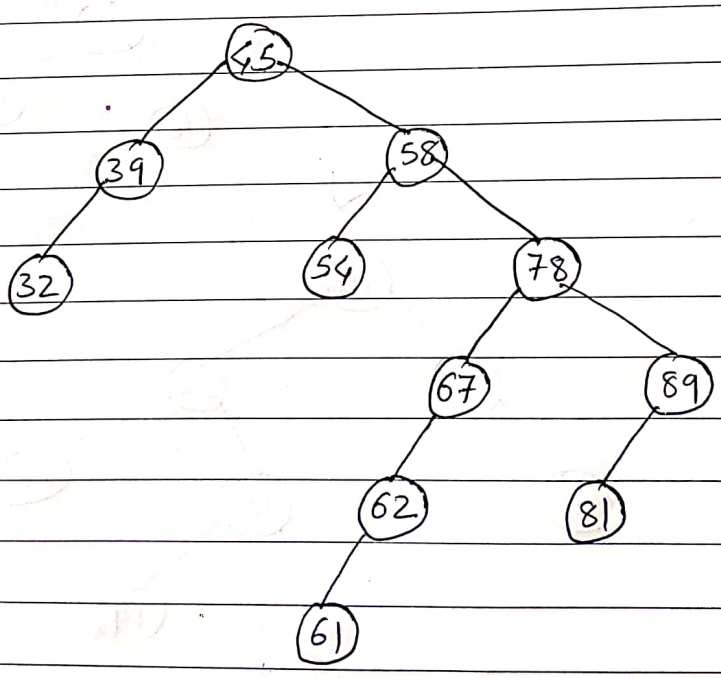




Case 4:- Deleting a node having 2 children

eg)





→ In ins
a red
check
the
→ In d
black
we c
the

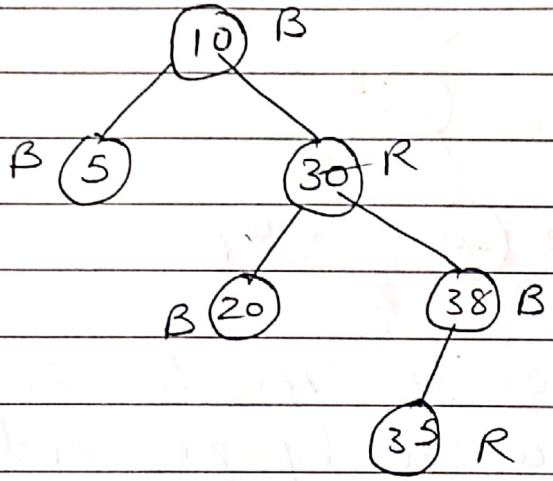
eg) De

→ In insert operation, we check if there is a red-red parent-child relationship. If yes, we check the color of parent's sibling to decide the appropriate case.

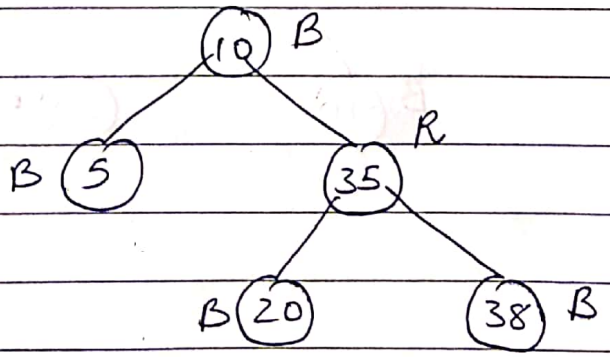
~~→ In delete operation, we check color of parent's~~

→ In delete operation, we check the count of black nodes in each path. If it is not same, we check the color of the sibling to decide the appropriate case.

eg) Deletion of a red node.



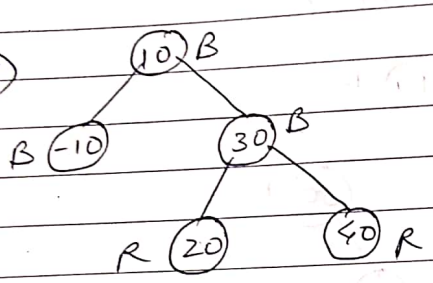
Delete 30. Its inorder successor is 35.



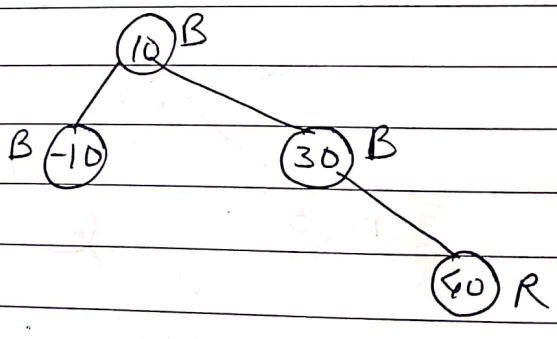
* Concept of double black node →
→ If the deleted node is black, the count of black nodes is violated.
→ The removal of a black node x causes any path that contains x, to have one less black node.

→ Two cases arise:-
1) The replacing node is red, in which case, we color it as black to cover the loss of one black node.

eg)



Delete 10. Its inorder successr is 20 so replace 10 by 20 and then delete 20. Here 20 is also a red node.



2) The repl
double

eg)

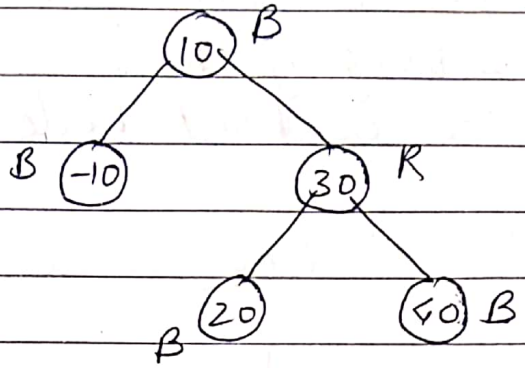
Delete
Its
20
node
doub

Th
to

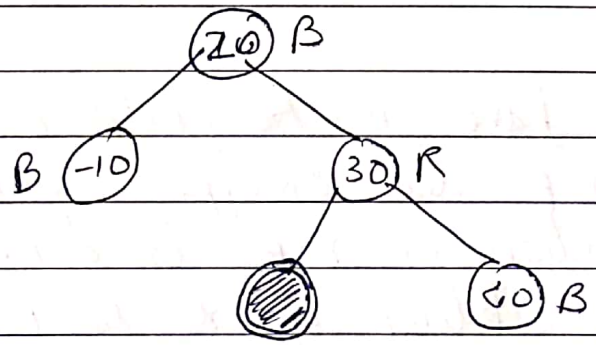
→ To
t

2) The replacing node is black, which creates a double black node.

eg)



Delete 10.
Its inorder successor is 20. So replace 10 by 20 and delete 20. Since 20 is a black node, ~~it creates~~ deleting 20 creates a double black node.



double black node.

This double black node needs to be removed to make the given tree as RBT.

→ To remove this double black node, consider there are four cases.

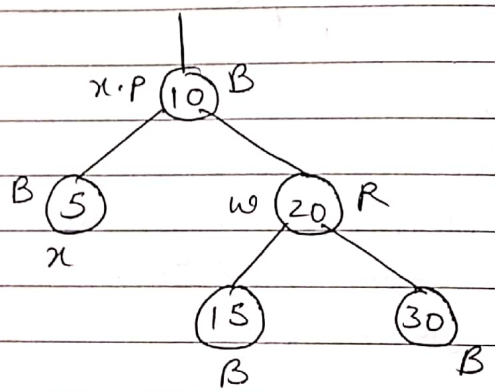
Smita Attarde

Date
Page 134

Terms used in all 4 cases:-

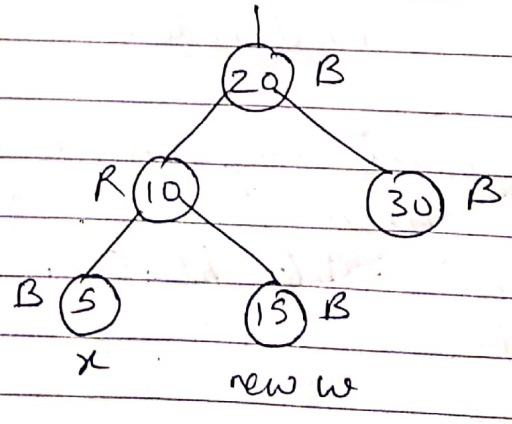
1) x → double black node.
2) $x.p$ → parent tree of x.
3) w → sibling of x.
4) c → red or black node.

Case 1:- x's sibling w is red.



→ Since w has black children, the colors of w and x.p are swapped and then left-rotation on x.p is done.

→ The new sibling of x is now black, and thus case 1 is converted to case 2, 3 or 4



case 2

c

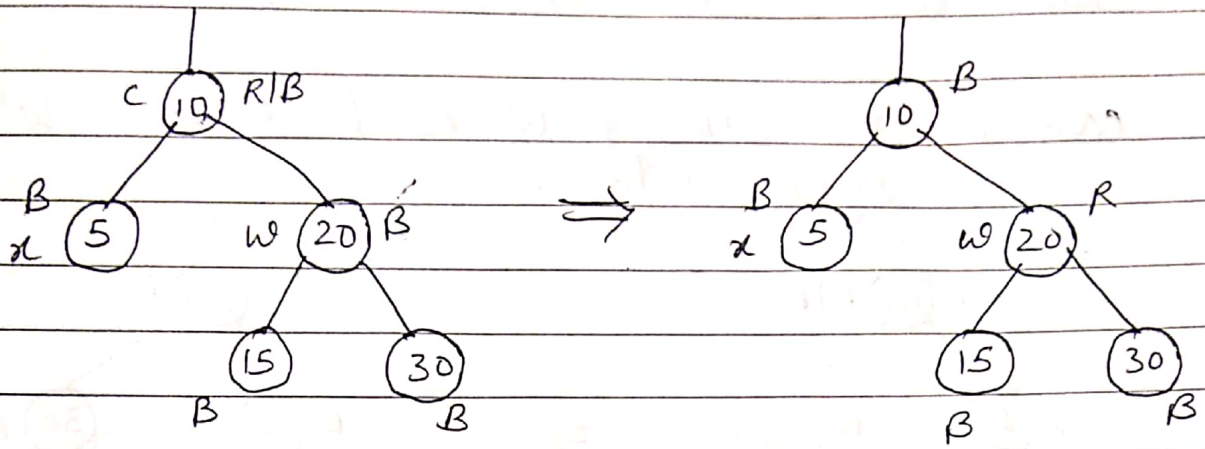
B
x
5

→ Since
make
→ So
we
whic

case

→ S
i

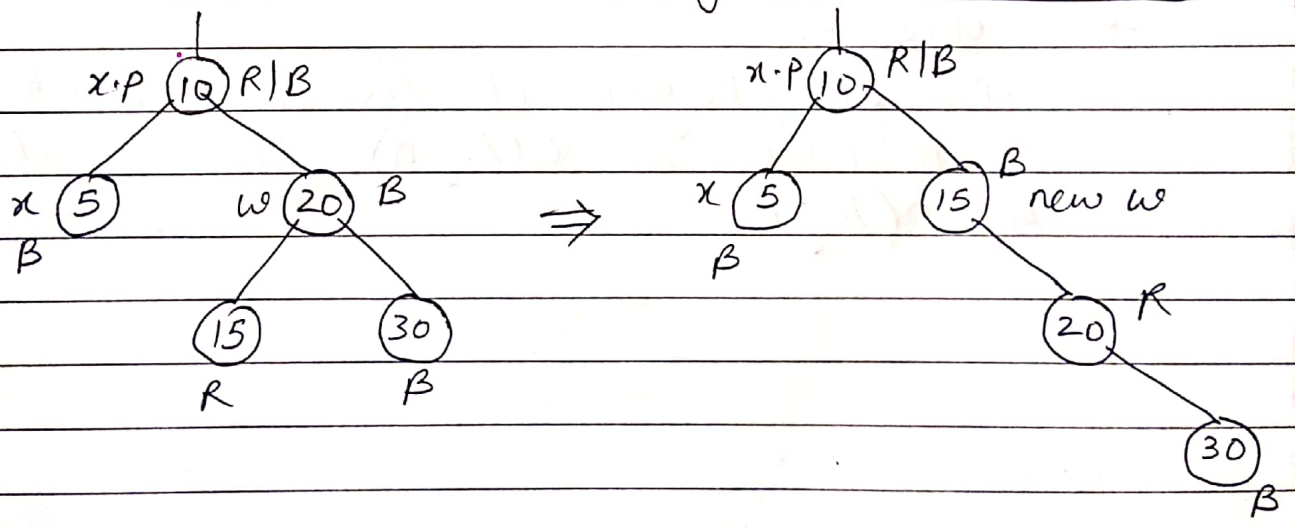
case 2:- x's sibling w is black and both of w's children are black.



→ Since w is also black, we remove it and make it as red.

→ To compensate for removing one black, we need to add one extra black to x.p which was originally either red or black

case 3:- x's sibling w is black, w's left child is red and w's right child is black.



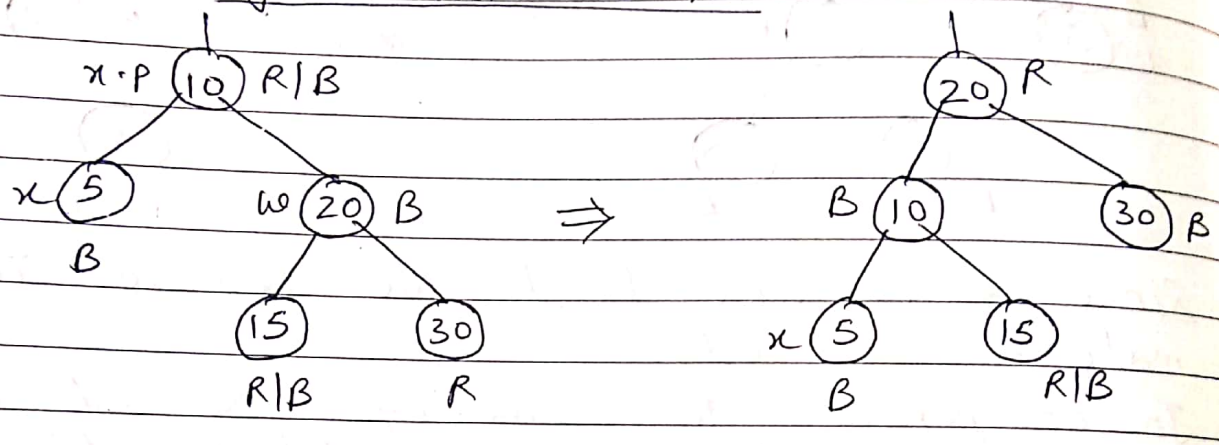
→ Swapping of colors is done between w and its left child w.left, and then right rotation on w is performed.

Smita Attarde

The new sibling w of x is now a black node with a red right child and thus case 3 is transformed to case 4.

Dec-18 10M

Case 4:- x's sibling w is black and w's right child is red.

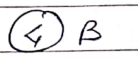


By making some color changes and performing a left rotation on x.p, we can remove the extra black on x, making it singly black.

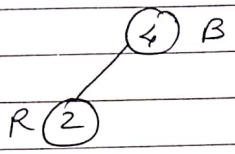
Analysis →
Since the height of the red black tree of n nodes is O(log n), the total cost is O(log n)

Dec-18 10M Create a red black tree for the following elements- 4, 2, 8, 10, 18, 6, 12, 14

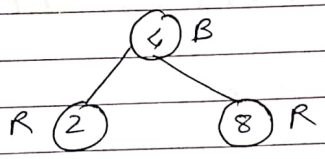
step 1 :- 4 →



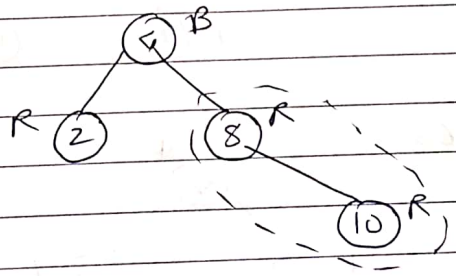
step 2 :- 2 →



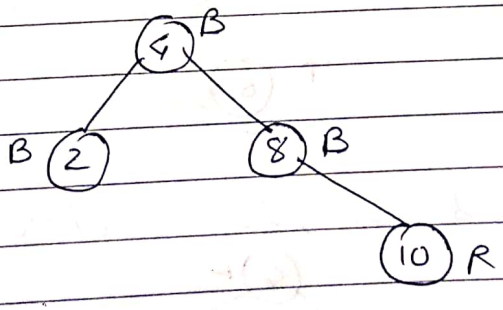
step 3 :- 8 →



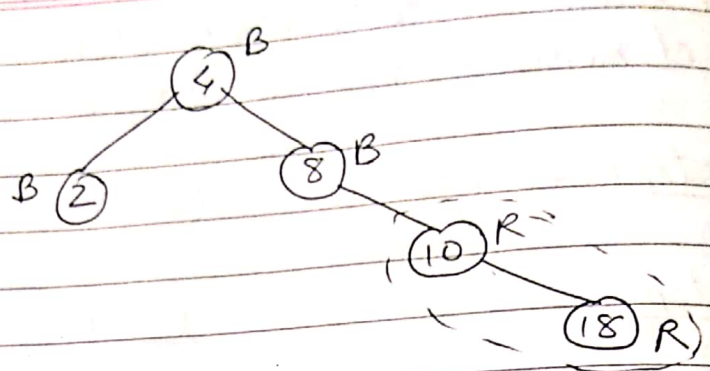
step 4 :- 10 →



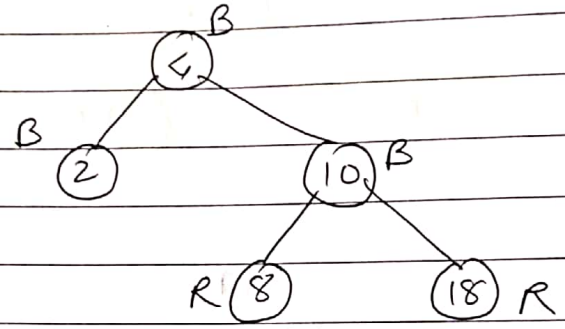
sibling is red, so recolor



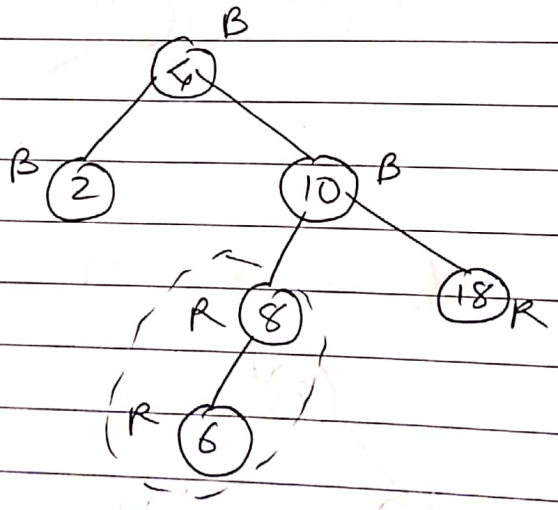
step 5:- 18→



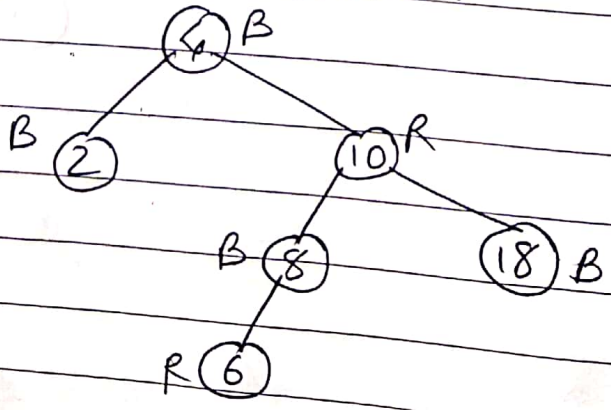
sibling is absent, so RR rotation



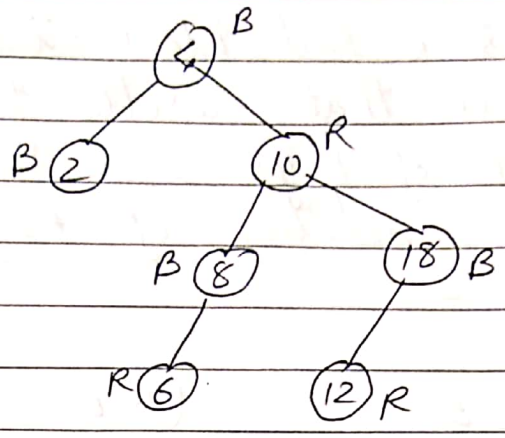
step 6:- 6→



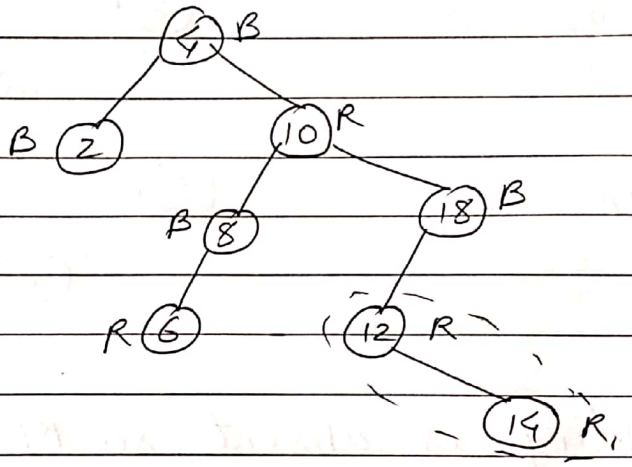
sibling is red, so recolor



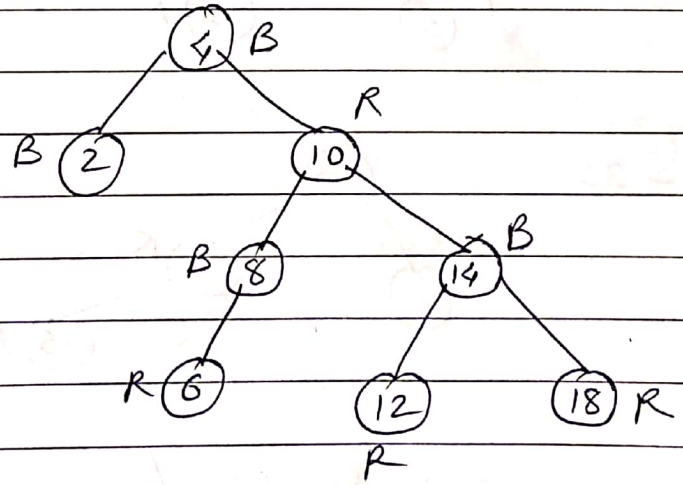
step 7:- 12→



step 8:- 14→



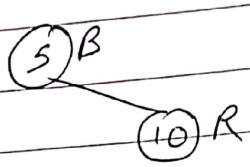
sibling is absent, so LR rotation.



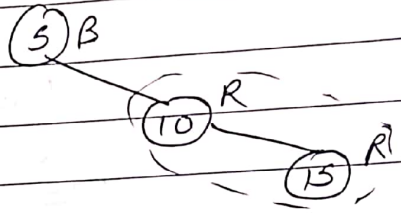
May-19 10M What is a red black tree? Show a red-black tree that results from successive insertion of keys 5, 10, 15, 25, 20, 30 and succesive deletion of keys 30, 25, 20, 15, 10 and 5

Sol^n:- step 1:- 5

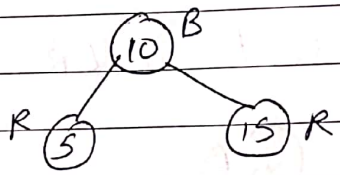
step 2:- 10



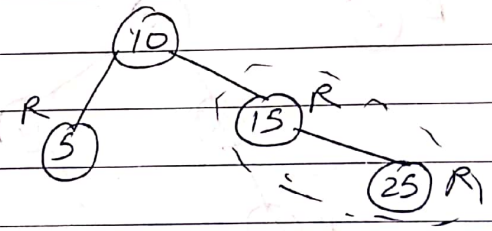
step 3:- 15



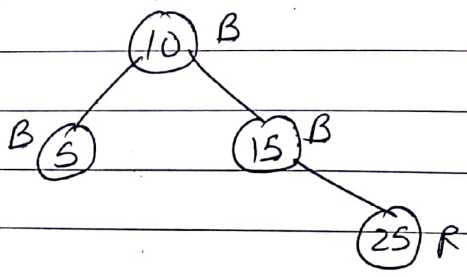
sibling is absent, so RR rotation.



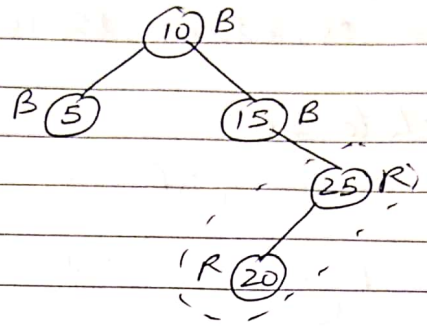
step 4:- 25



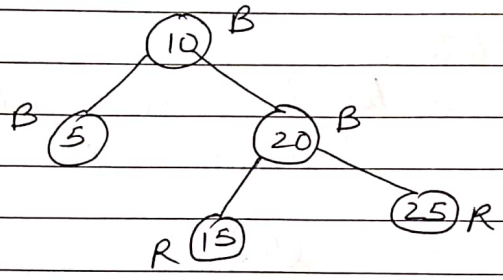
sibling is red, so recolor



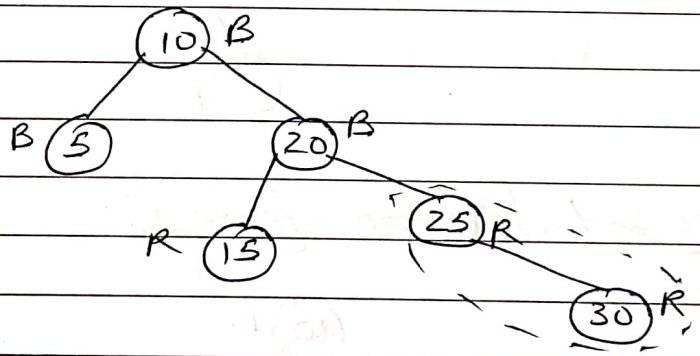
step 5:- 20



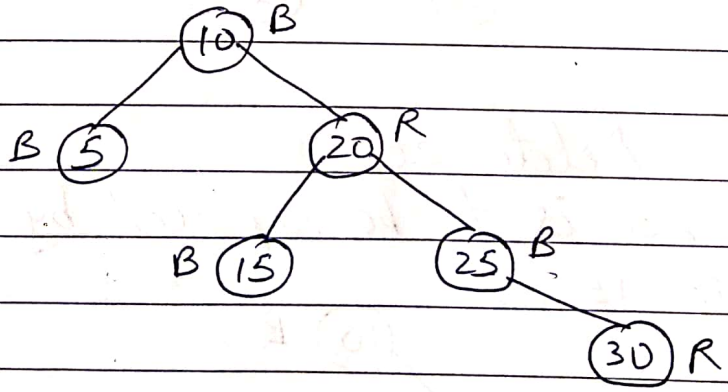
sibling is absent, so RL rotation.



step 6:- 30

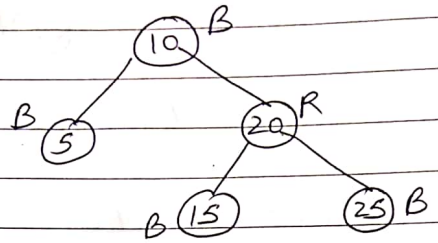


sibling is red, so recolor



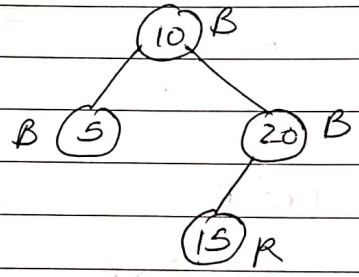
Deletion → 30, 25, 20, 15, 10, 5

step 1:- Delete 30

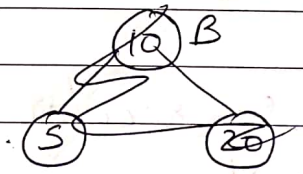


step 2:- Delete 25

Here, 25 is black & its sibling is black with 2 black children (null pointers). So case 2

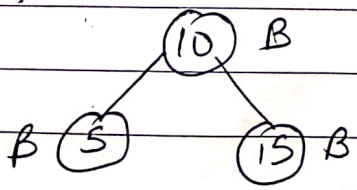


~~step 3:- Delete 20 [case 2]~~



step 3:- Delete 20

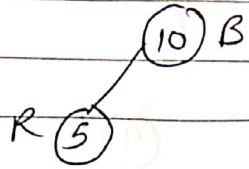
Here, 20 is to be replaced by 15 & then delete 15.



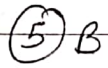
Dec-16
10M.

Soln.:-

step 4:- ~~Delete~~ Delete 15 [case 2]



step 5:- Delete 10

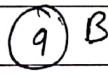


step 6:- Delete 5

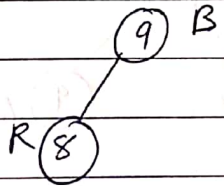
No ~~Tree~~ Tree.

Dec-16 10M. What is red black tree? Show the red-black tree that results from the successive insertion of the following keys 9, 8, 7, 3, 5, 2 and the successive deletion of the following keys 2, 5, 3, 7, 8, 9.

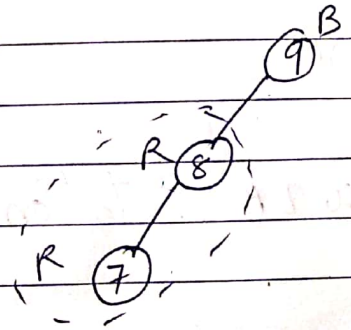
Sol":- step1:-9 →



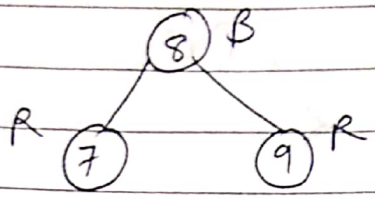
step 2:- 8



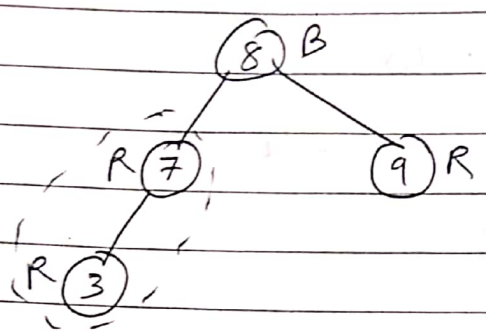
step 3:- 7



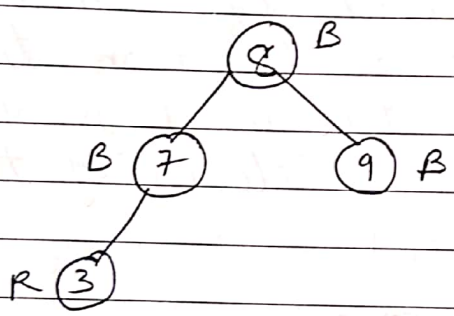
sibling is absent, so LL rotation



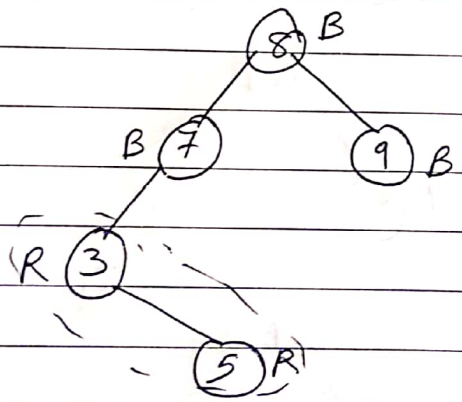
step 4:- 3



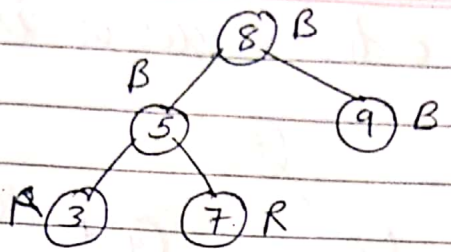
sibling is red, so recolor.



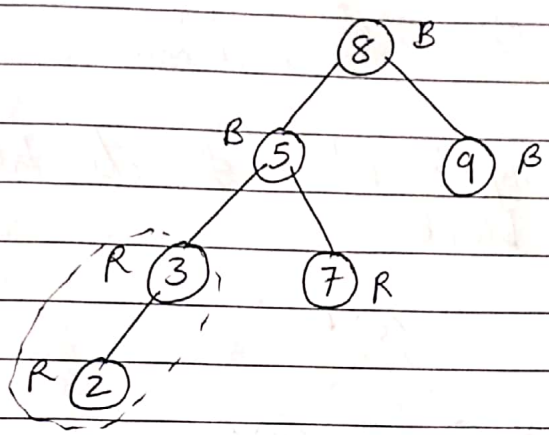
steps:- 5



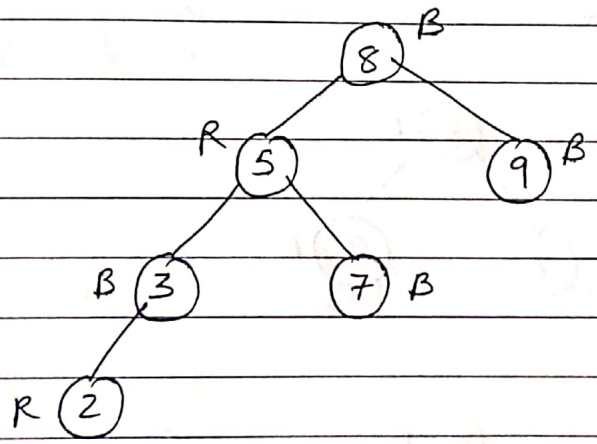
sibling is absent, so LR rotation



step 6:-2

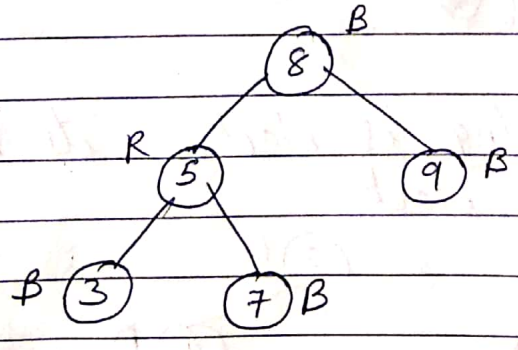


sibling is red, so recolor.

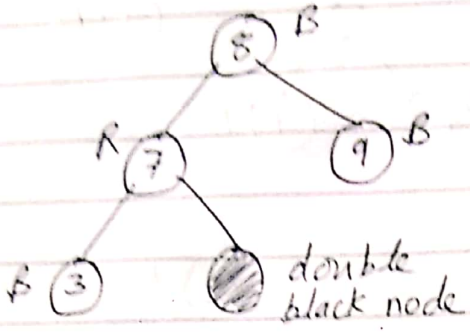


Delete 2, 5, 3, 7, 8, 9.

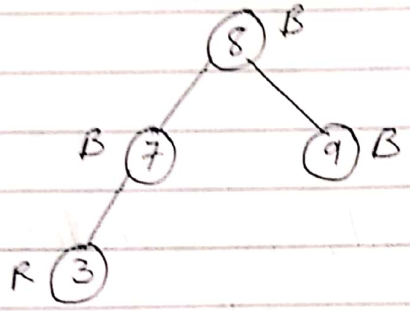
Delete 2



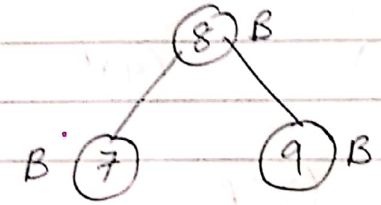
Delete 5 : Inorder successor is 7



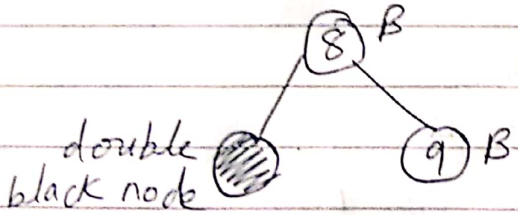
case 2: sibling is black & its both children are black.



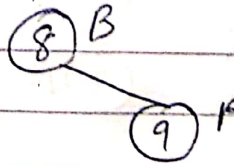
Delete 3:



Delete 7:



case 2: sibling is black & its both children are black.



Delete 8-

(9) B

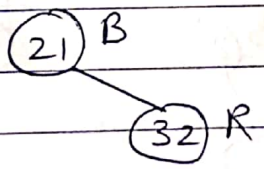
Delete 9: Empty Tree.

May 17 ~~10M~~ 12M Explain Red-black tree. Show the red-black tree that results after each of the integer keys 21, 32, 64, 75 and 15 are inserted, in that order, into an initially red-black tree. Delete keys 75 and 32 from the created tree. Clearly show the tree that results after each insertion and deletion (indicating the color of each node), and make clear any rotations that must be performed.

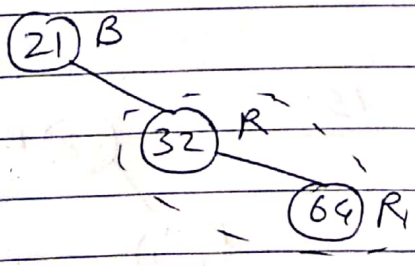
Soln: Insert 21, 32, 64, 75, 15

step 1: 21

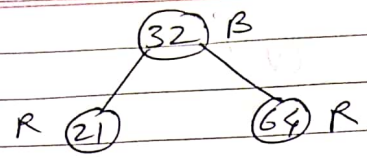
step 2: 32



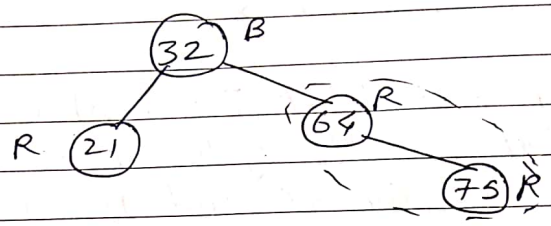
step 3: 64



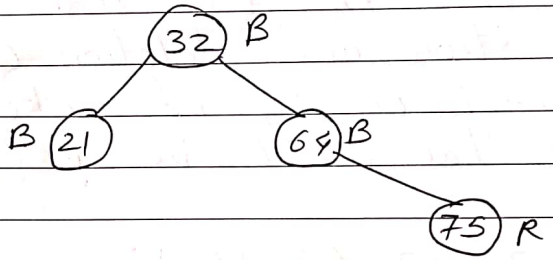
sibling is absent, so RR rotation.



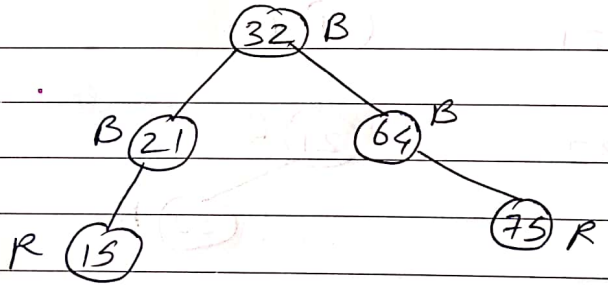
step 4: 75



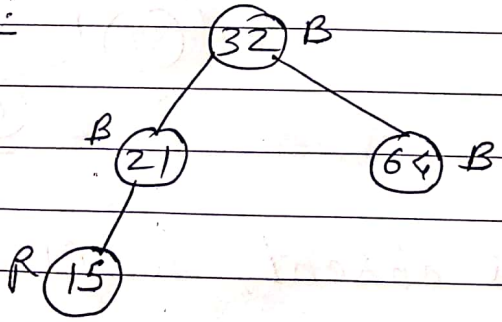
sibling is red, so recolor



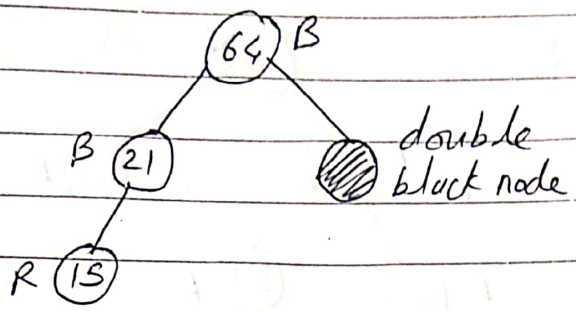
step 5: 15



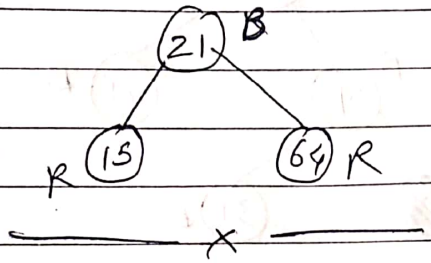
Delete 75:



Delete 32: inorder successor is 64.

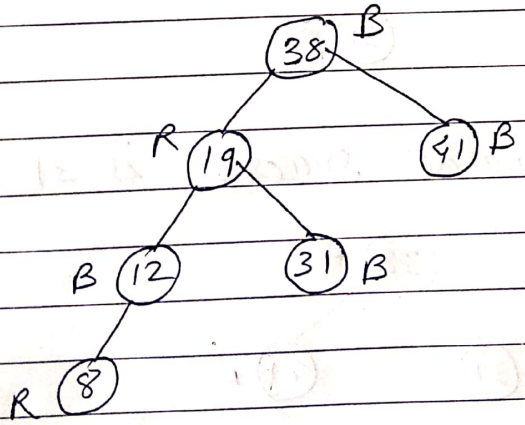


case 3: sibling is ~~red~~ black & its left child is red and right child is black.



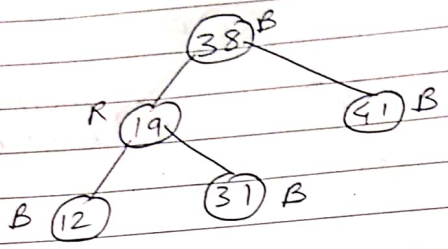
4-16 OM What is RBT? Show the RBT that results from the successive insertion of the following keys 41, 38, 31, 12, 19, 8 and the successive deletion of the following keys 8, 12, 19, 31.

→ RBT created for this elements on page-120.

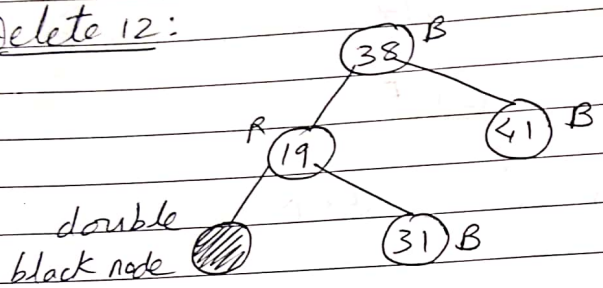


Delete 5, 12, 19, 31

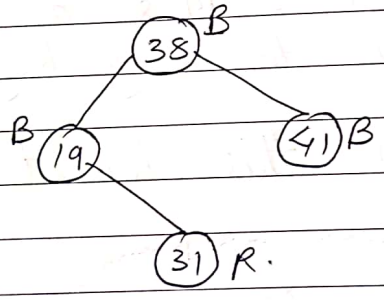
Delete 5:



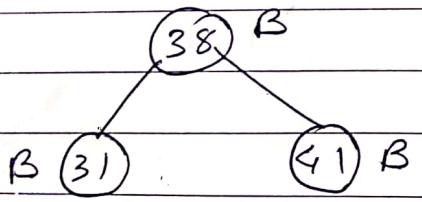
Delete 12:



case 2: sibling is black & its both children are black.



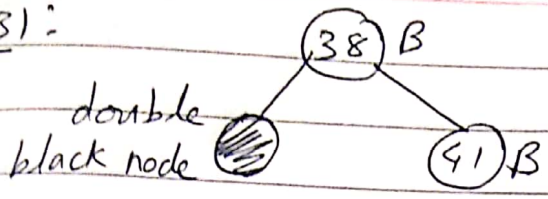
Delete 19: inorder successor is 31.



Delete 31

case 2:

Delete 31:



case 2: sibling is black and its both children are black.