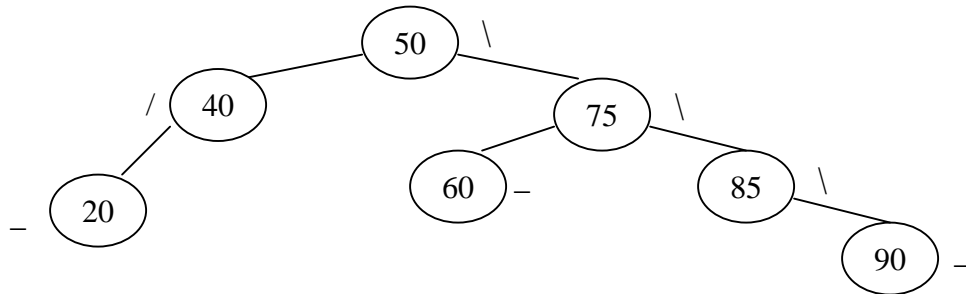
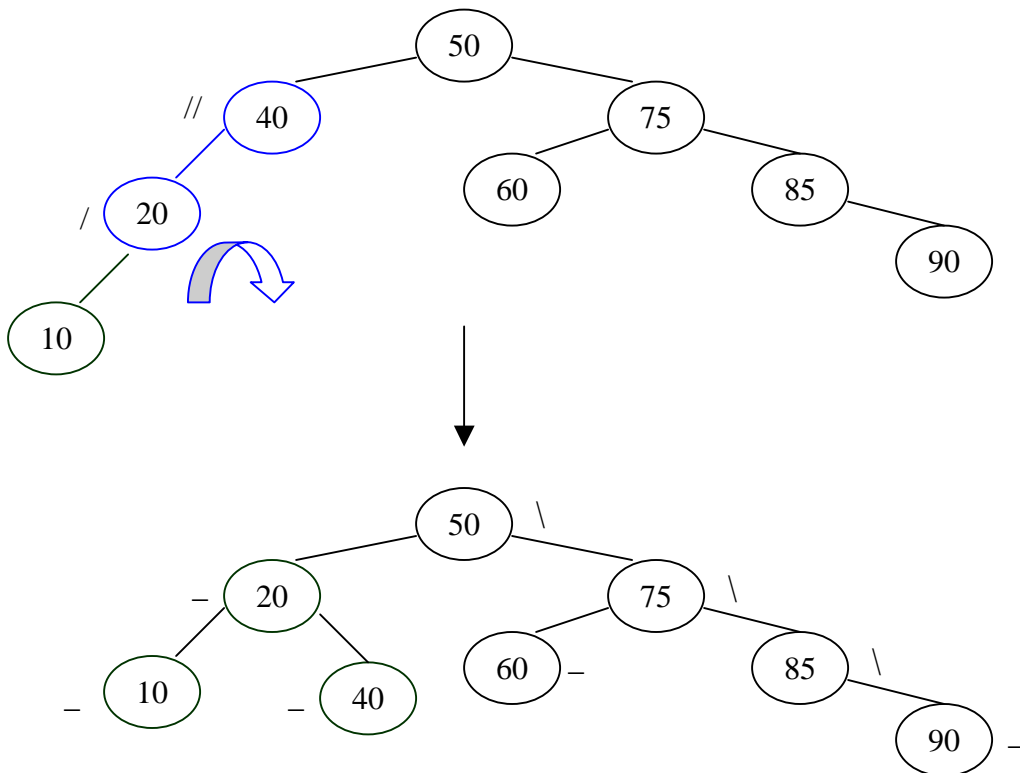


1)



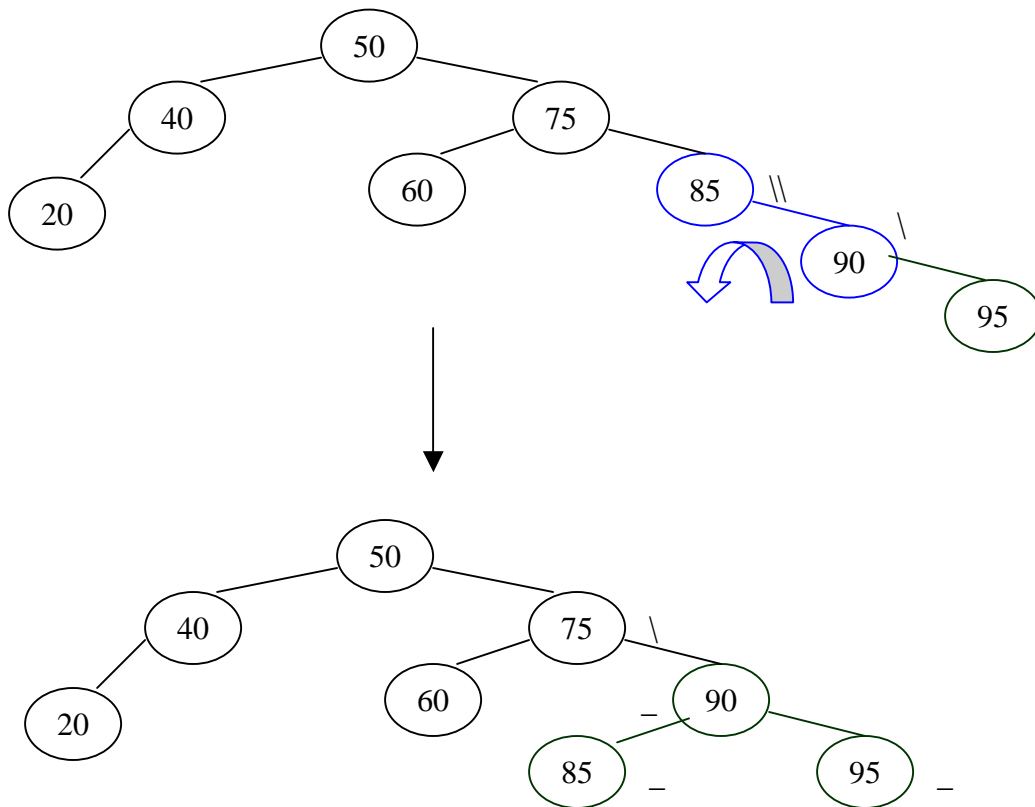
2) Tree with insertion of key 10:

Key 40 is left higher and its left sub tree is left higher too. So we can balance the tree with one right-rotation only !

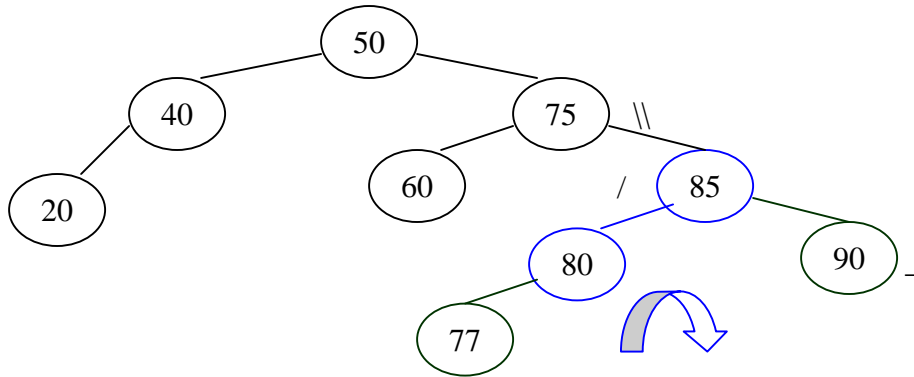


3) Tree with insertion of the key 95:

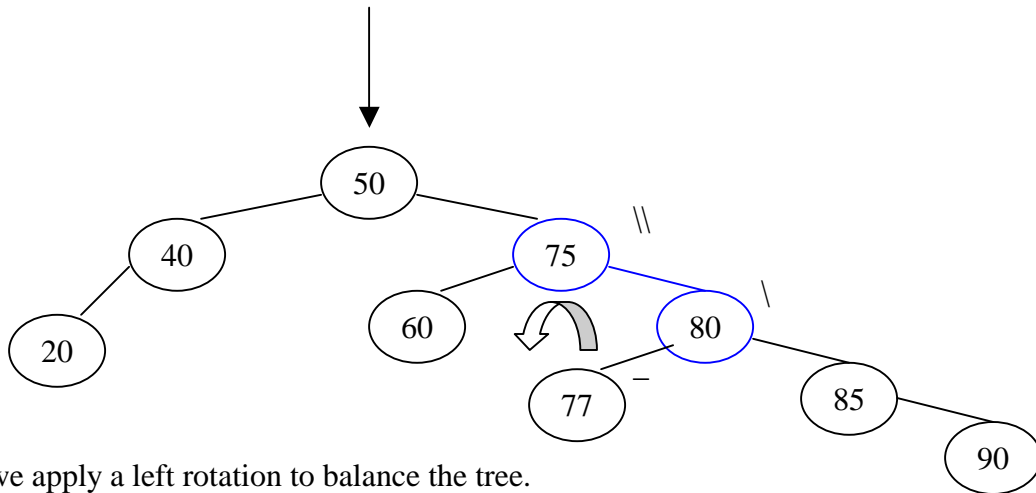
The key 85 has become right higher with a right sub tree right higher also. We balance the tree with a left rotation.



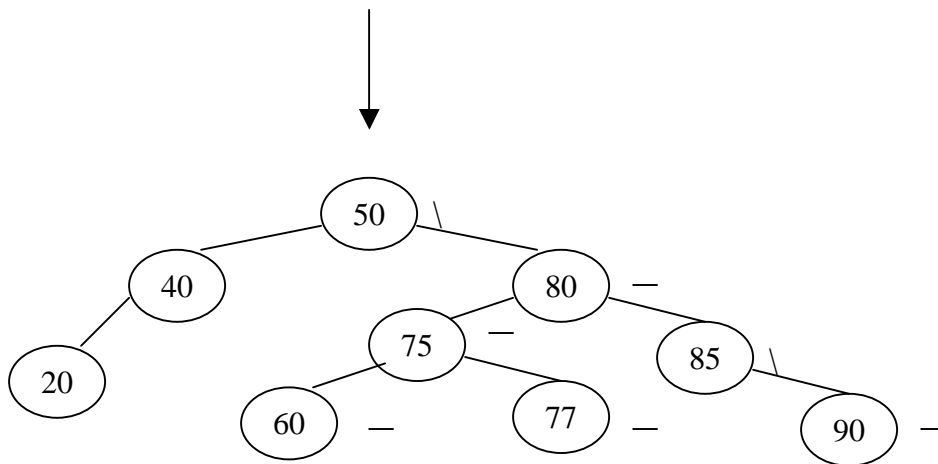
- 4) Tree with insertion of key 80 and 77.
 The key 75 is right higher with a right sub tree left higher!
 We need a double rotation.



First we make the right sub tree become right higher with a right rotation

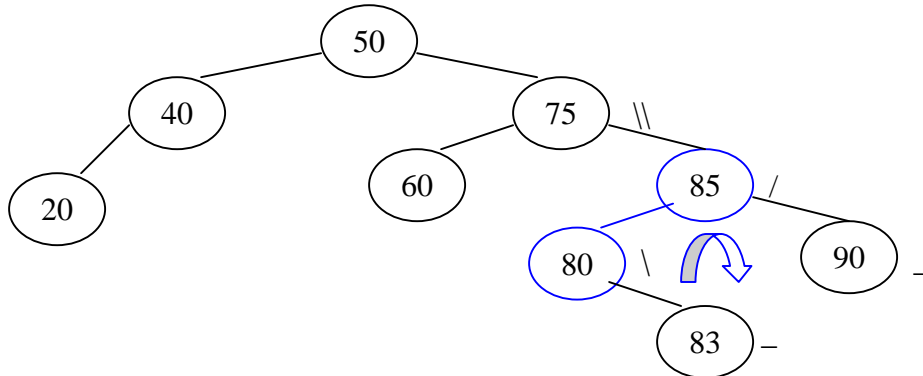


Then we apply a left rotation to balance the tree.

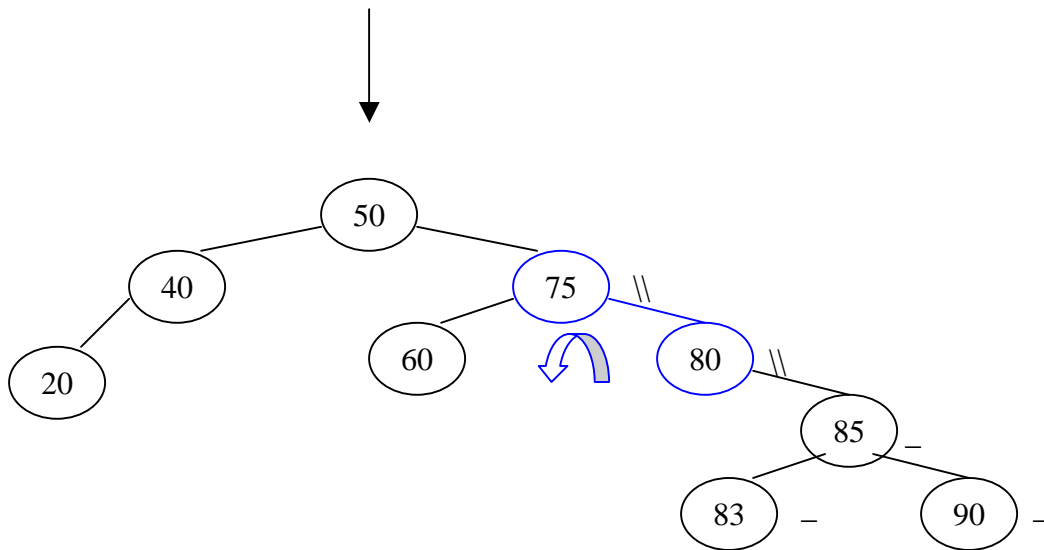


5) Tree with insertion of the key of key 80 and 83

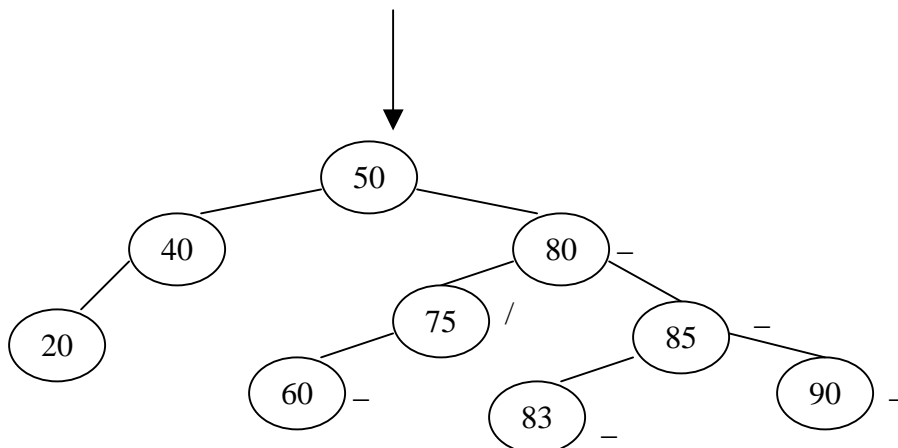
The key 75 is right higher again with a sub tree left higher. We need 2 rotations!



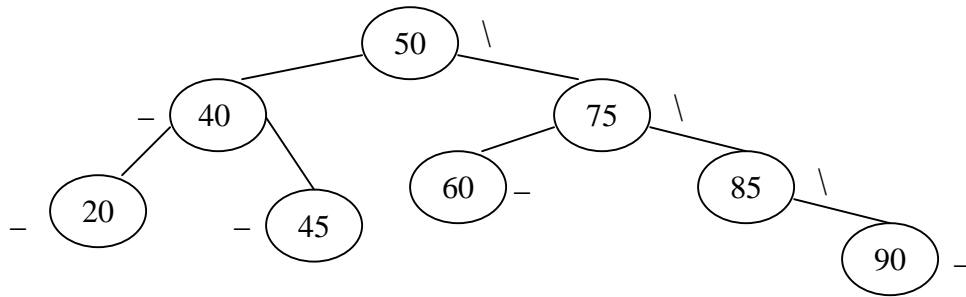
We make the sub-tree become right higher with a right-rotation.



A left rotation is applied at the imbalanced node (75).

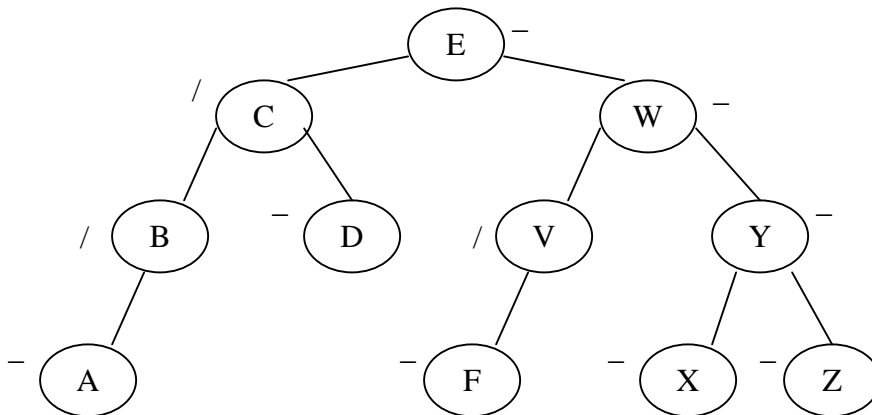


- 6) Tree with insertion of key 45:
No rotations needed!



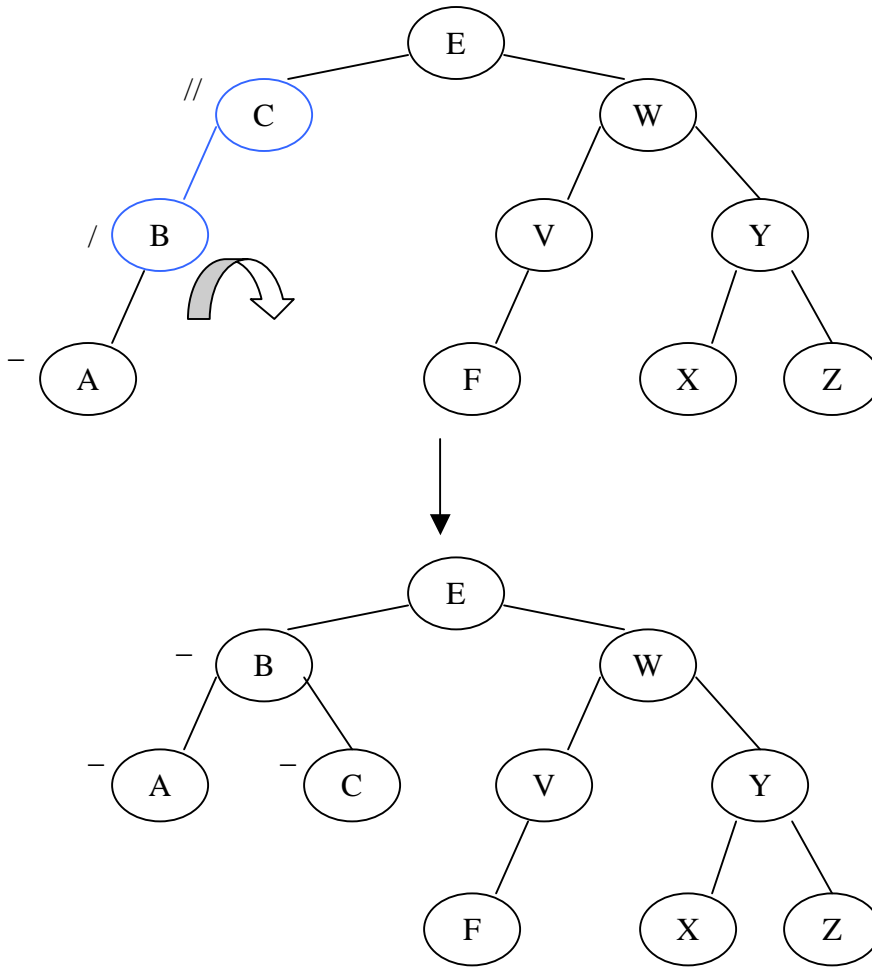
7)

Starting tree labeled:



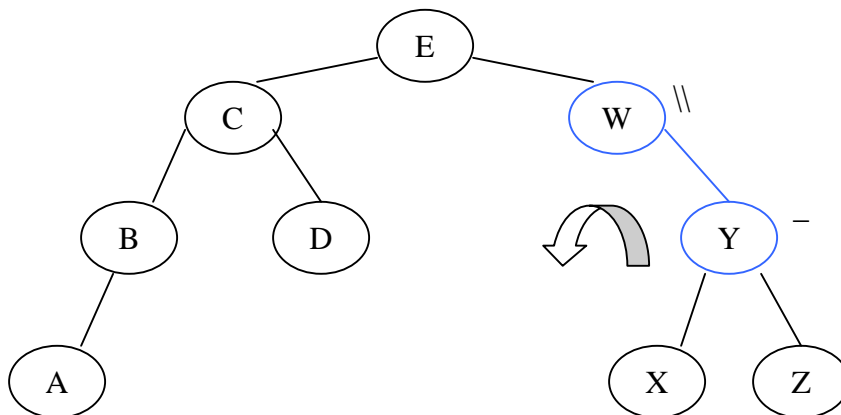
Tree with deleted key D:

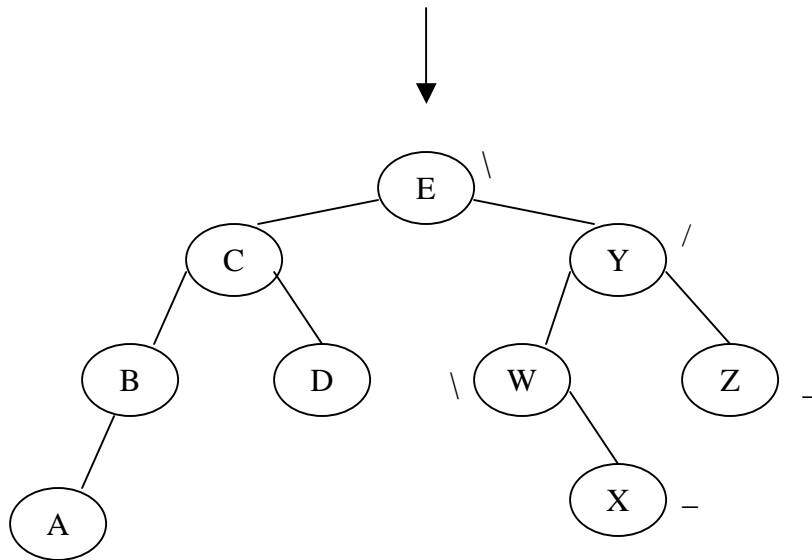
Key C is left higher with a left higher sub-tree. One right rotation is enough to balance the tree.



8) Tree with deleted key F and V:

The key W is right higher with an equally balanced sub-tree. We apply a left-rotation.

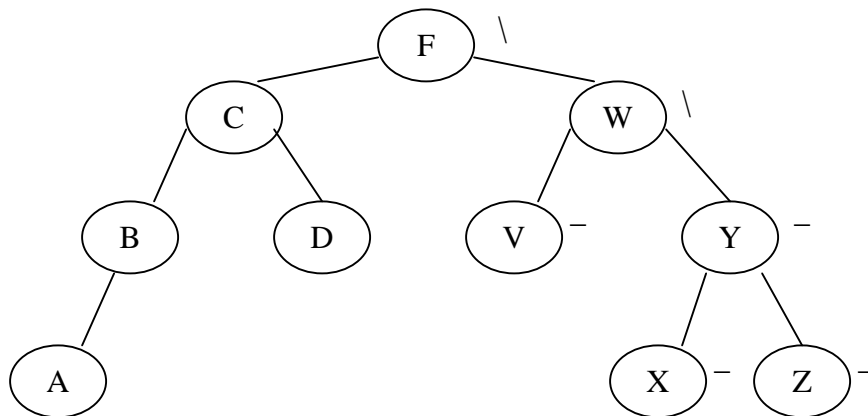




9)

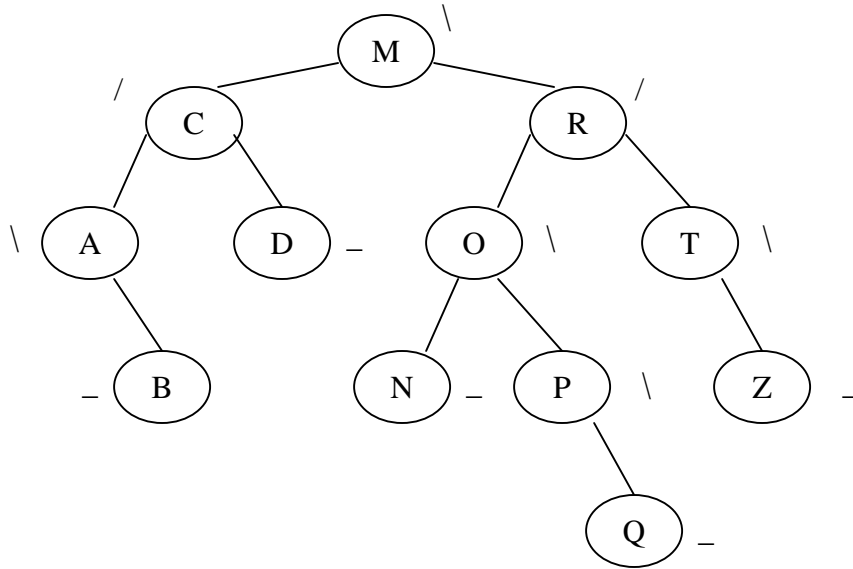
Tree with deleted key E: Deletion of the root

We search for the minimum value node in the right sub-tree. It's F in our case. We then swap F with E! No rotation is needed!



10)

Starting Tree labeled :



Tree with key T deleted:

The key R is left higher with a sub tree right higher. We need 2 rotations, a left one to make the sub tree left higher and a right one to balance the tree.

