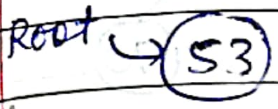
Binary Search Tree

Values to be inserted

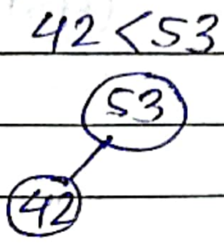
53, 42, 18, 11, 36, 24, 10, 62, 85, 9, 90, 52

Insertion:

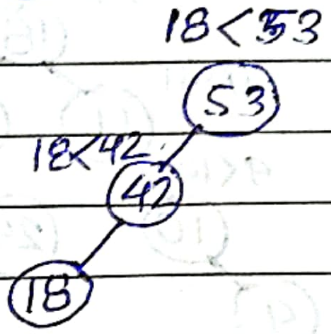
1. Num = 53 (Root)



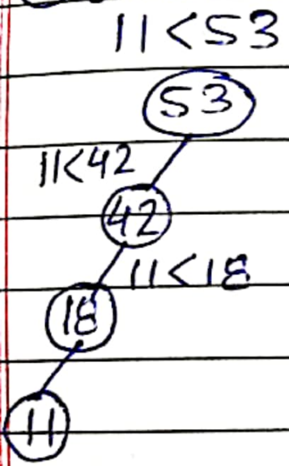
2. Num: 42



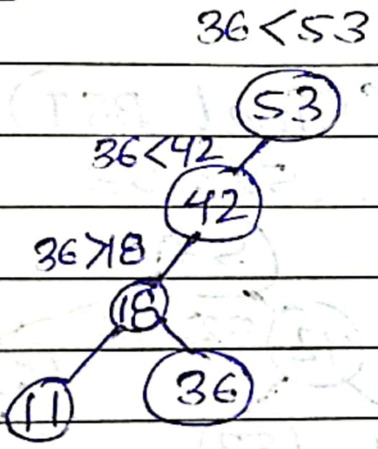
3. Num: 18



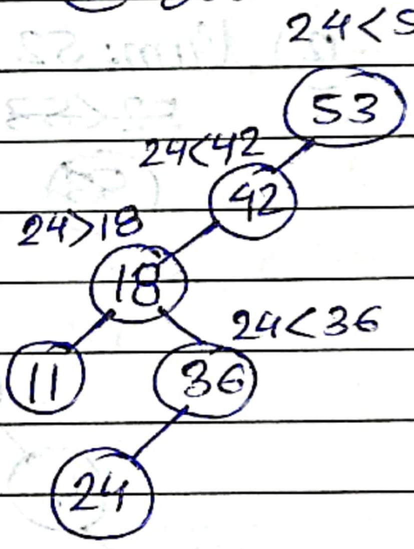
4. Num: 11



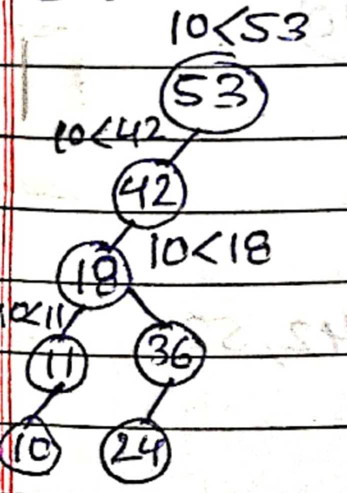
5. Num: 36



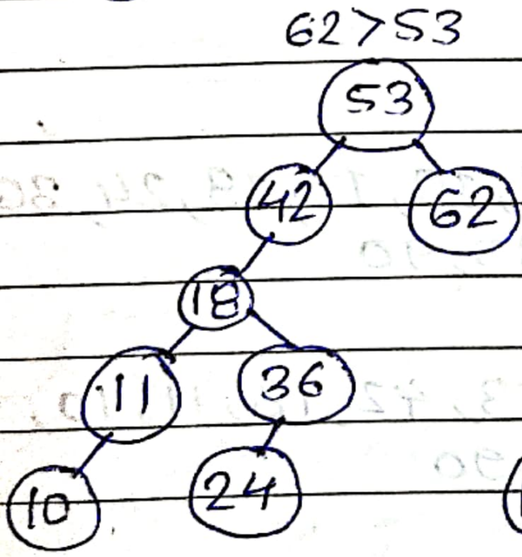
6. Num: 24



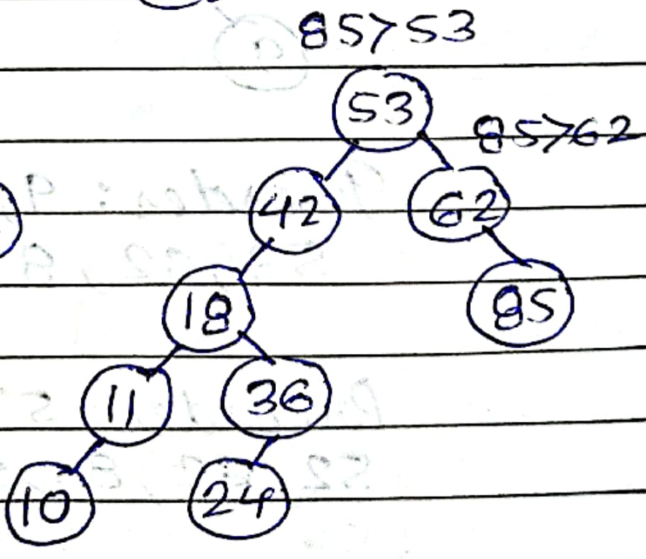
7. Num: 10



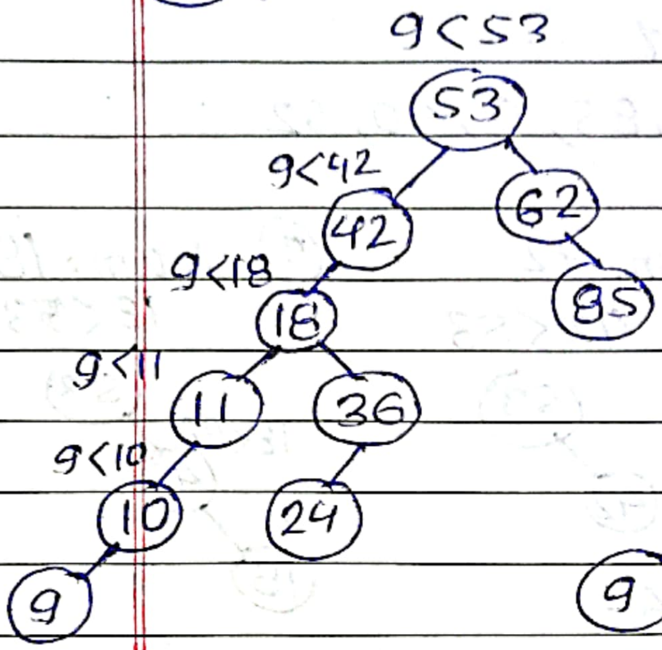
8. Num: 62



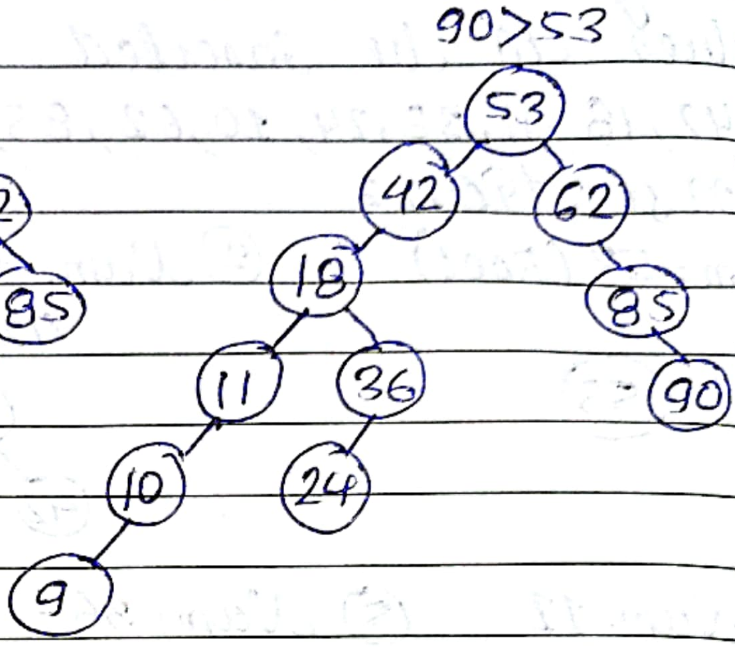
9. Num: 85



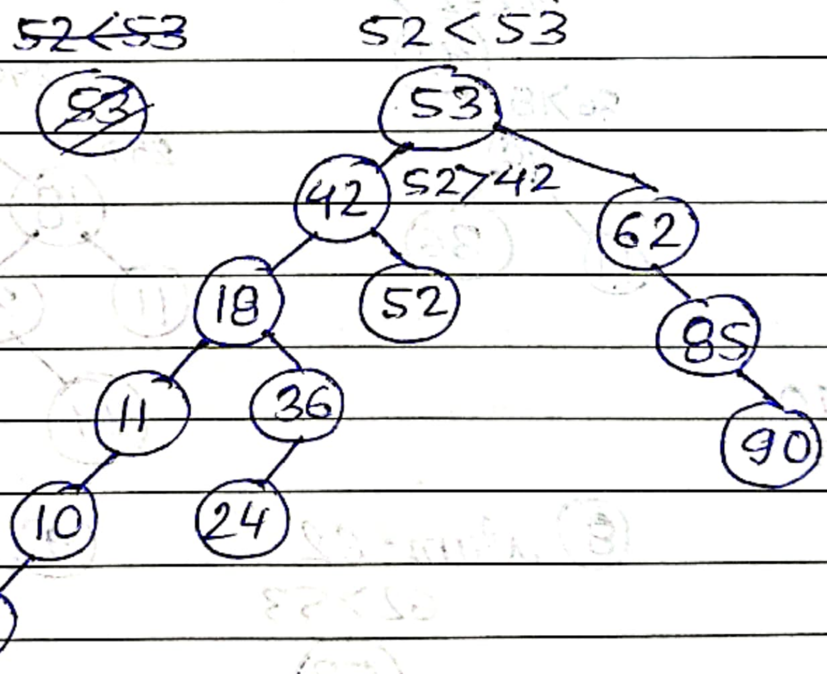
(10) Num: 9



(11) Num: 90



(12) Num: 52 (Final BST)



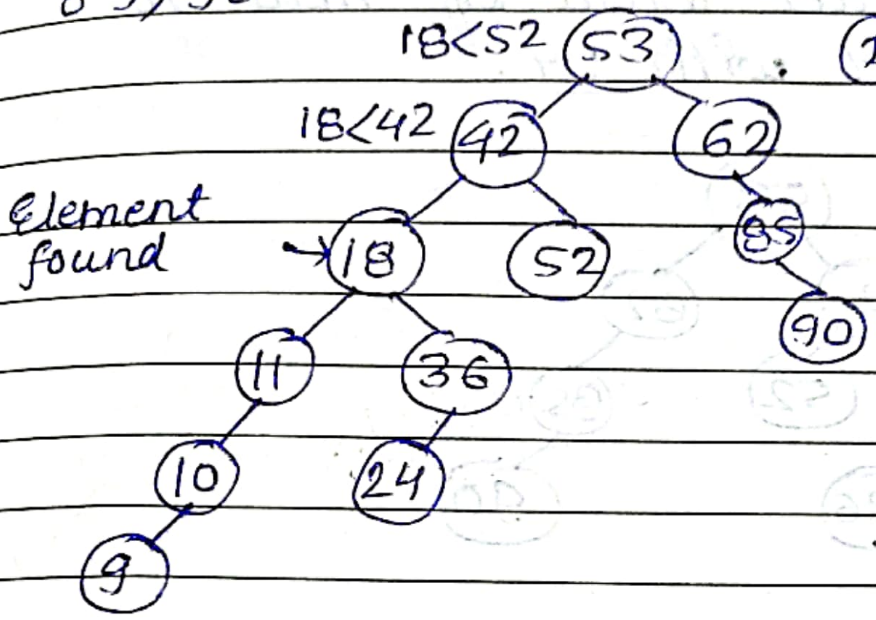
Inorder: 9, 10, 11, 18, 24, 36, 42, 52, 53, 62, 85, 90

Preorder: 53, 42, 18, 11, 10, 9, 36, 24, 52, 62, 85, 90

Postorder: 9, 10, 11, 24, 36, 18, 52, 42, 90, 85, 62, 53

Deletion: Element: 18

Inorder: 9, 10, 11, 18, 24, 36, 42, 52, 53, 62, 85, 90



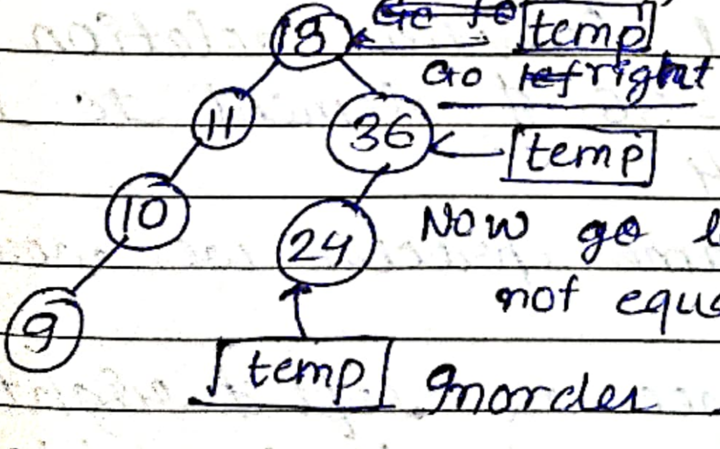
① First find the element

- As element is found, check the left & right subtrees of the element.
- We can see neither left nor right subtrees of given element are NULL
- Now find Inorder Successor

Inorder Successor: Element which succeeds a given element in Inorder Traversal.

→ It is the leftmost element of the right subtree of the given element

⇒ Consider the tree from 18:



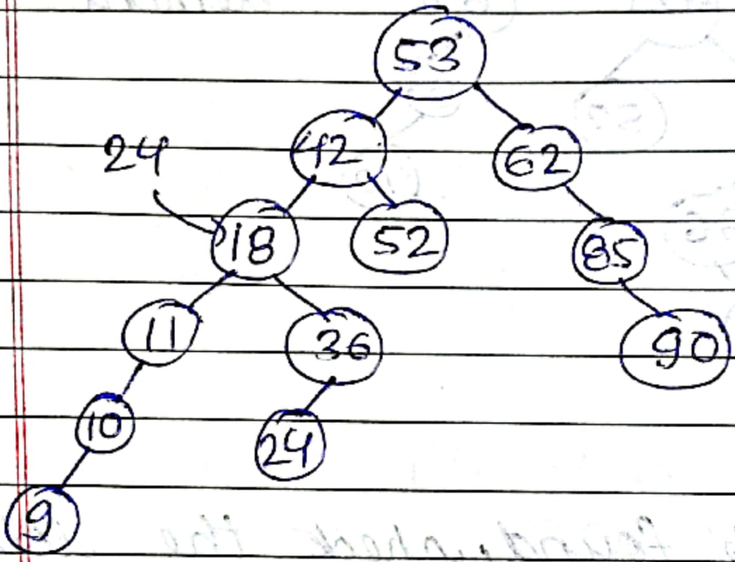
Take a temporary pointer temp pointing to 18

Now go left until temp→left not equal to null

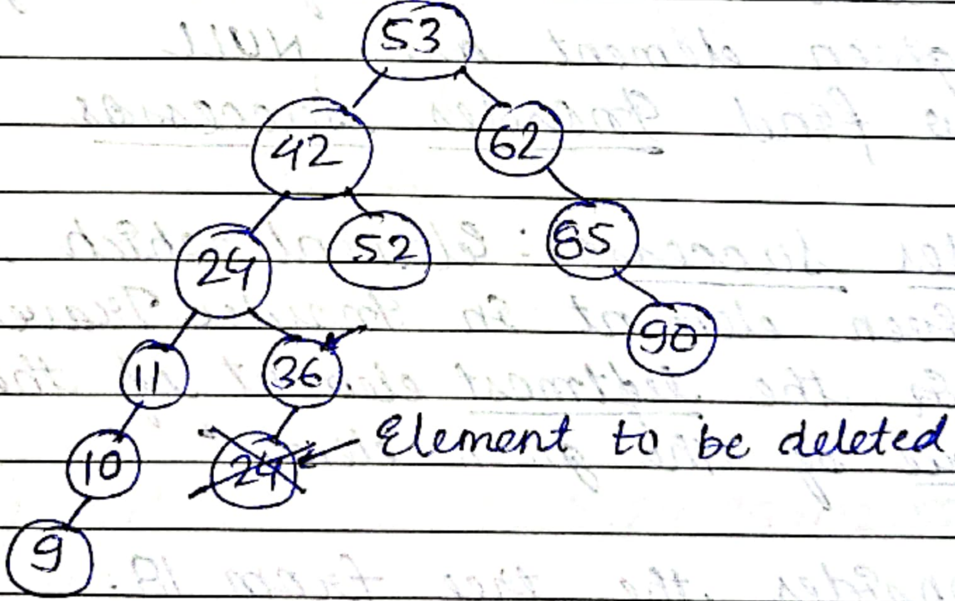
Inorder successor found

Inorder successor = 24

⇒ Now replace Data of node to be deleted with <u>24</u>.



View now:



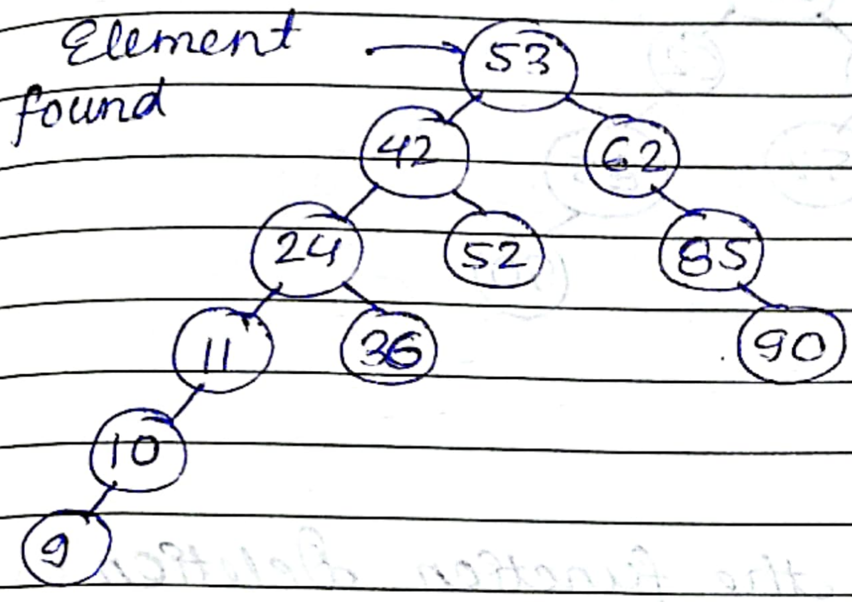
→ Now call function deletion with root = 36 & element to be deleted = 24

→ By the same process we would find 24

→ We can see as left & right both of the node to be deleted are NULL. Delete it

Deletion: Element=53 52,

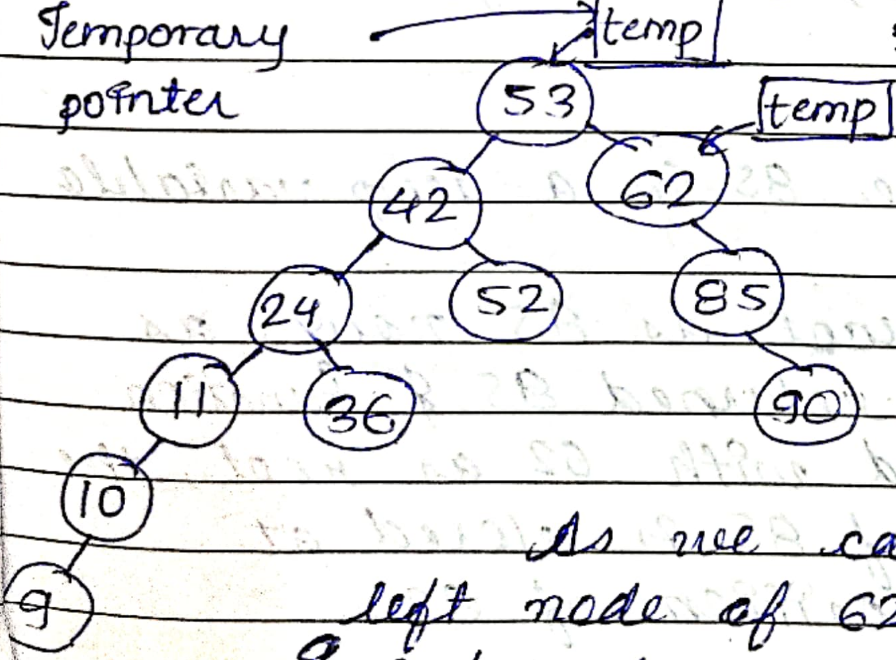
Inorder: 9, 10, 11, 24, 36, 42, 53, 62, 85, 90



1) search for the element

Left & right subtrees of element to be deleted are not null, not even one of them is null.

→ Find Inorder successor now.



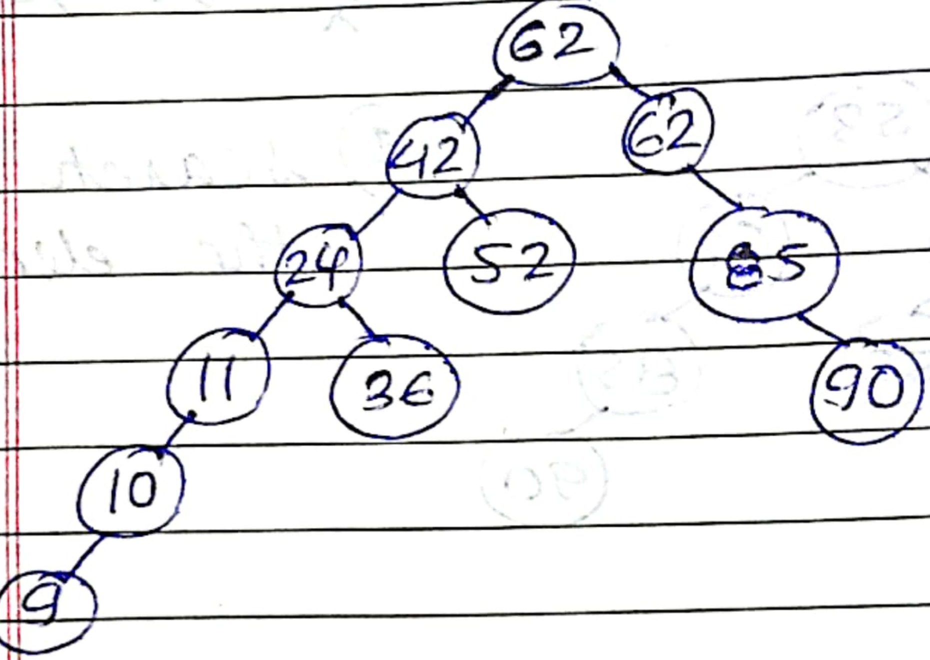
1) First go right

2) Now go left until temp→left is not null

As we can see the left node of 62 is null, Inorder successor is 62

→ Now replace 53 (Node to be deleted) with 62

Tree now:

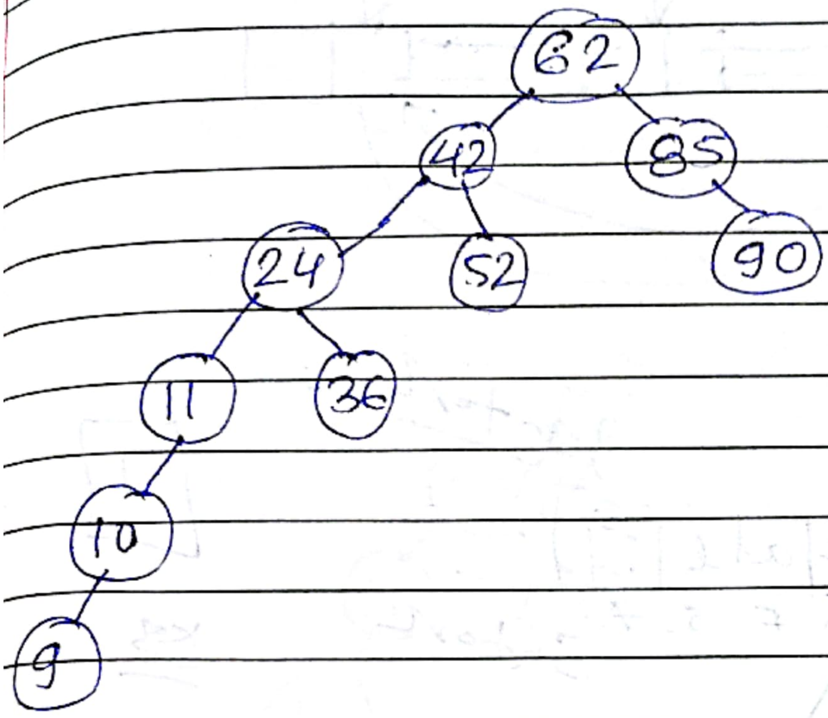


Now call the function Deletion with root = 62 & element to be deleted = 62

As the element to be deleted is root only, we have found the element.

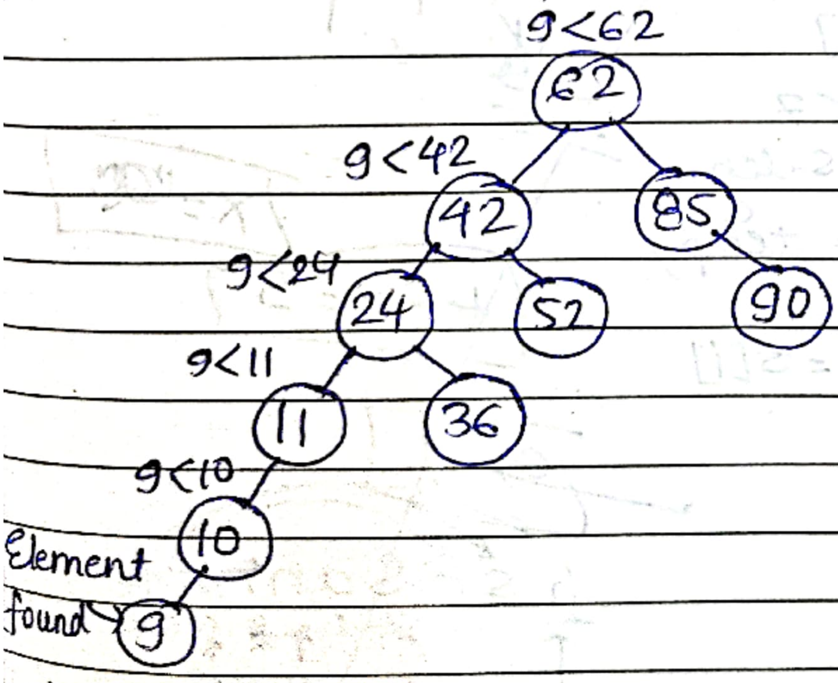
→ Now store 85 in a temp variable
→ Free 62
→ Return root as 85 now, as we have returned 85 & function was called with 62 as root, the position of 85 is stored at previous position of 62.

Tree now:



Inorder now: 9, 10, 11, 24, 36, 42, 52, 62, 85, 90

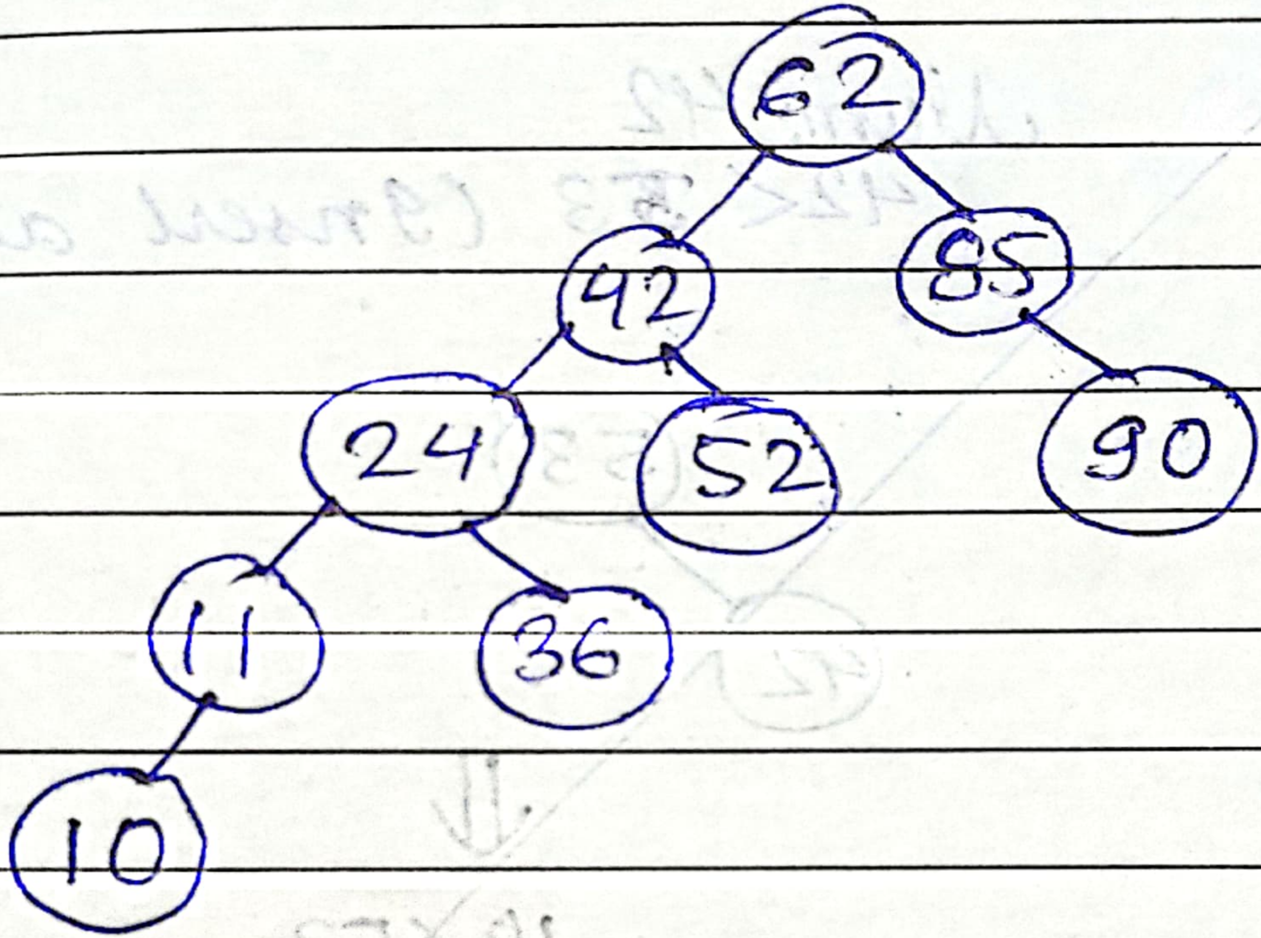
Deletion: Element = 9
Inorder: 9, 10, 11, 24, 36, 42, 52, 62, 85, 90



1) Search for element

→ As we can see both left node & right node of the element to be deleted are null.
→ Directly delete 9.

Inorder now : 10, 11, 24, 36, 42, 52, 62, 85, 90


62
42
85
24
52
90
11
36
10