AVL Tree.

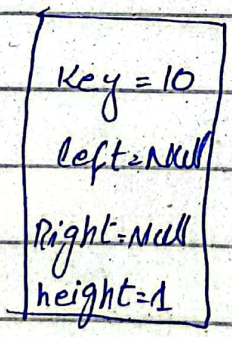
Insertion Dry Run:

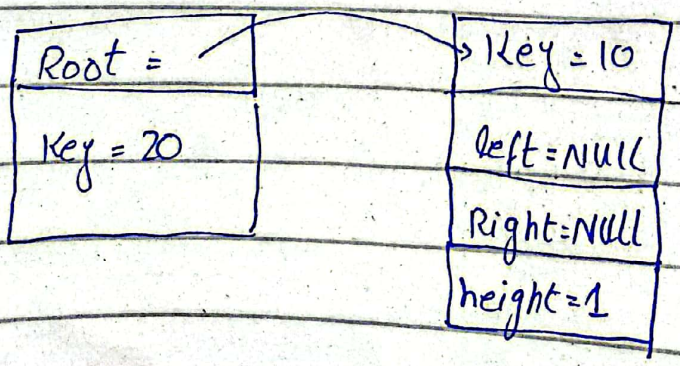
Root = NULL

Root = NULL

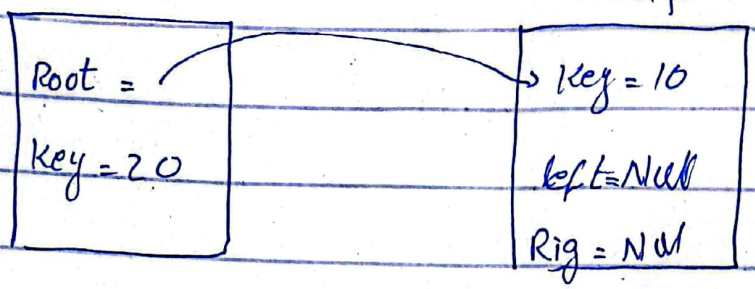
Key = 10

If root == NULL



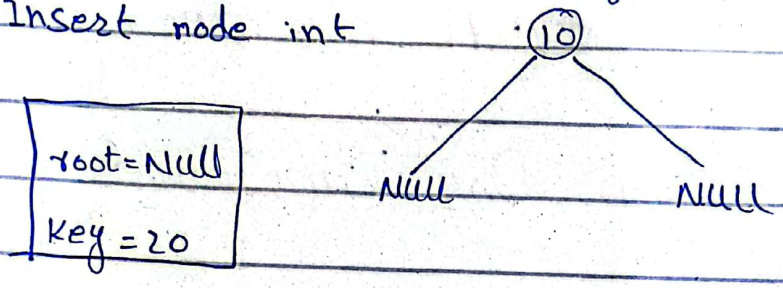


Heap

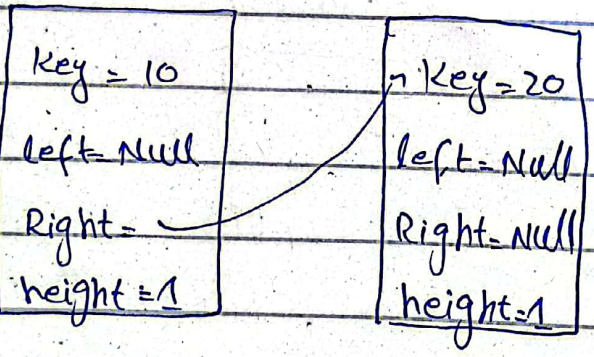


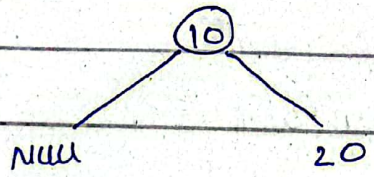
height = 1

Insert node int



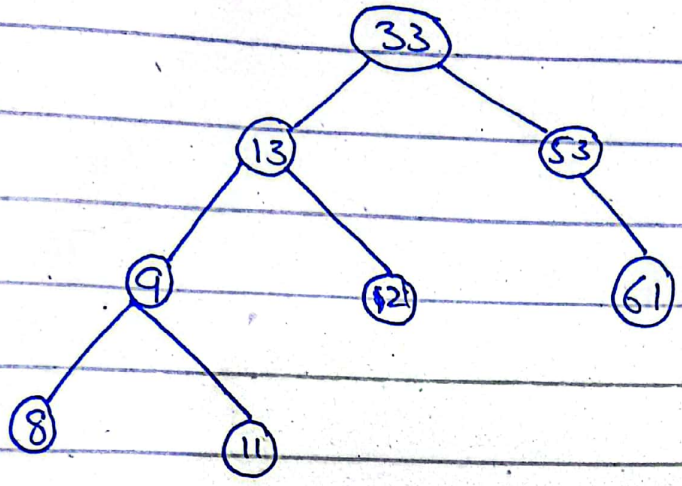
heap



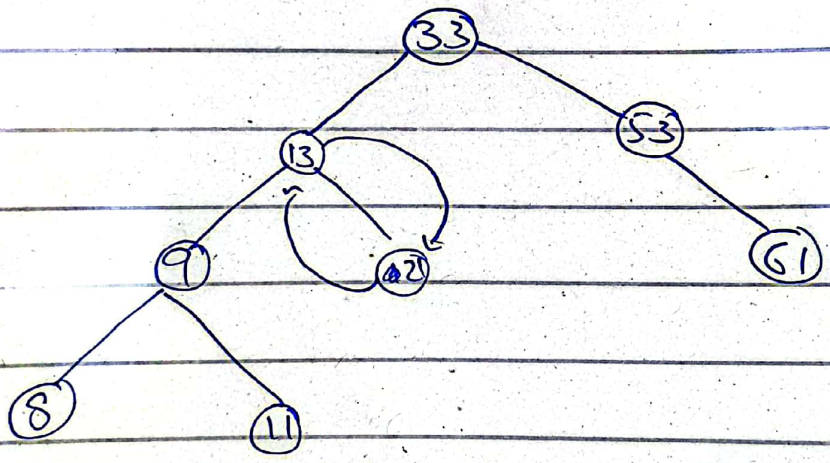


Deletion:

Let we have a AVL Tree



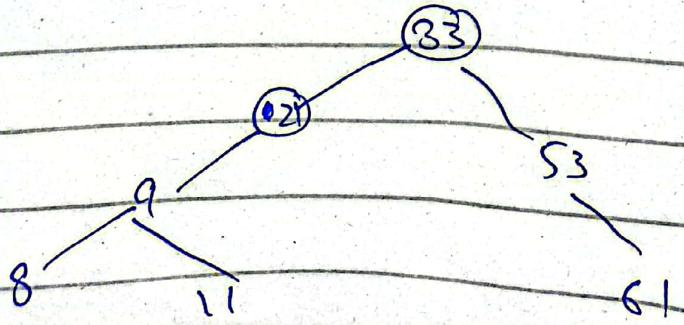
Delete node 13:



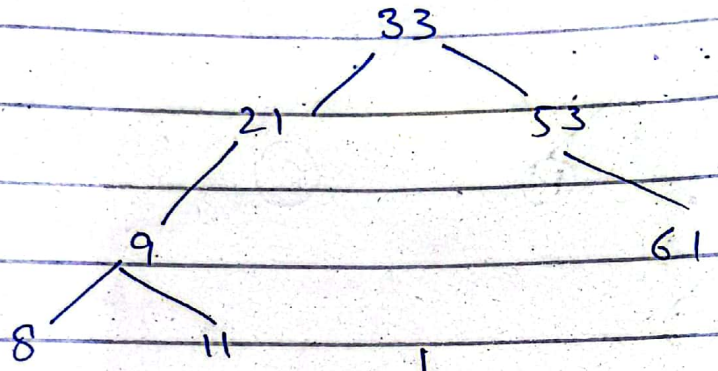
Swap the element with the edge node.

∴

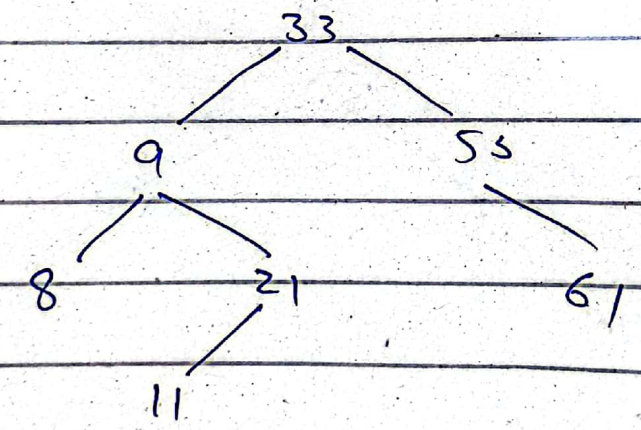
Now delete.



Now balance the 21



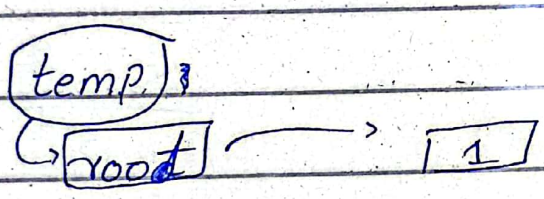
Final Tree.

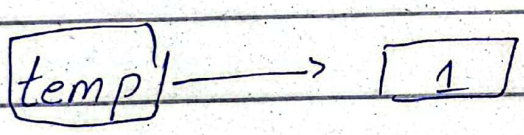


Name: M. Hassan.
Reg #: SP22-BCS-045
Section: (A)

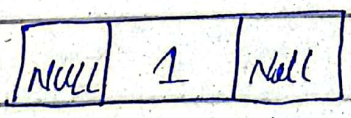
Pre Order traversal:

```
Node * temp = new node
```

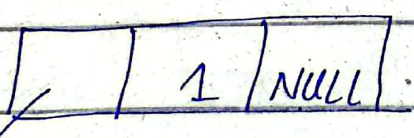


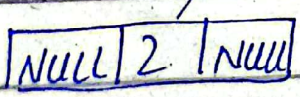



```
temp left = temp -> right = NULL;
```




```
root -> left = new Node (2);
```

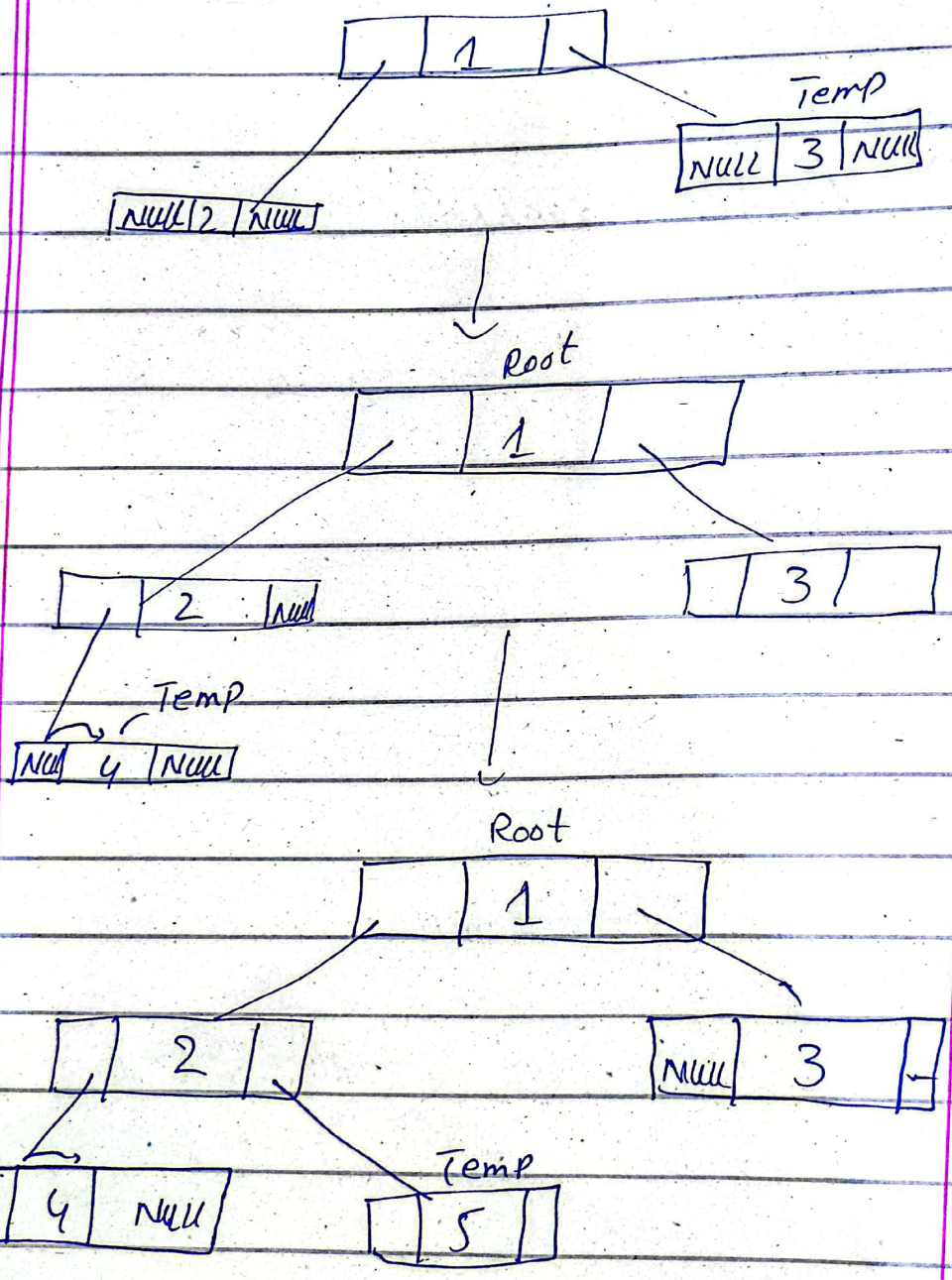




root -> right -> newNode(3);

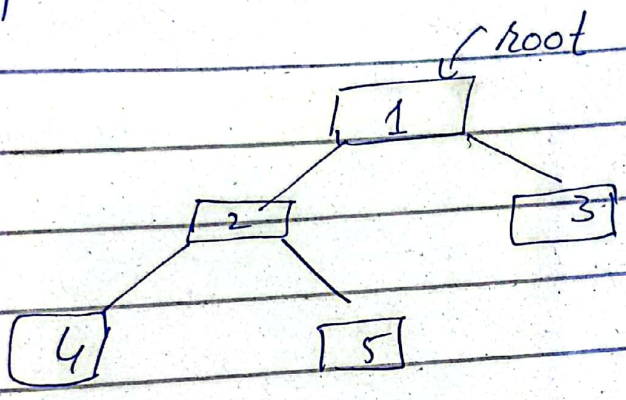
same procedure repeated

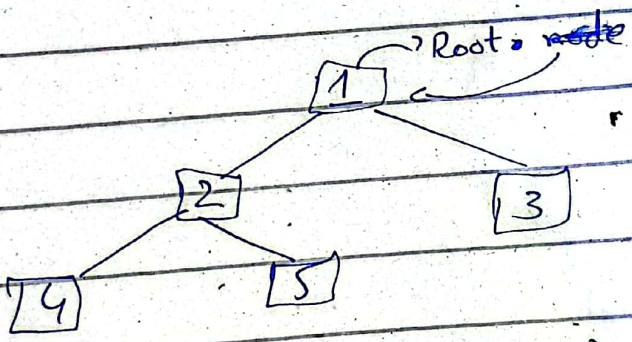
now temp will point in right



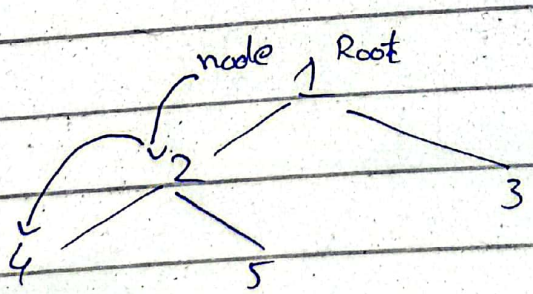
Post order tree traversal:

Tree is created as in previous code.



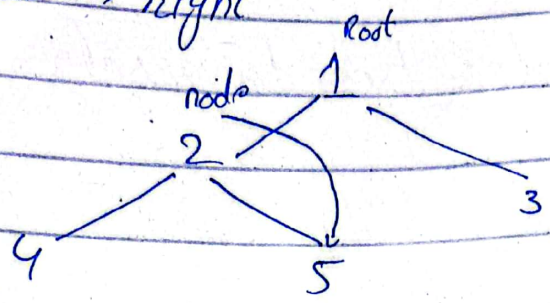


Post order (node → left)

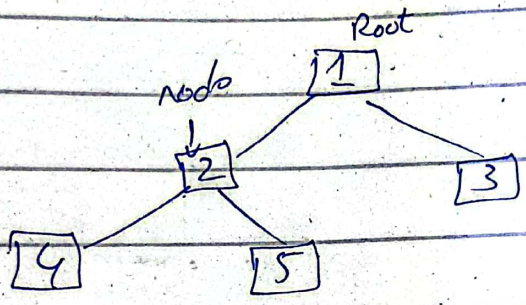


4 has no further child so it is print.

Node → right

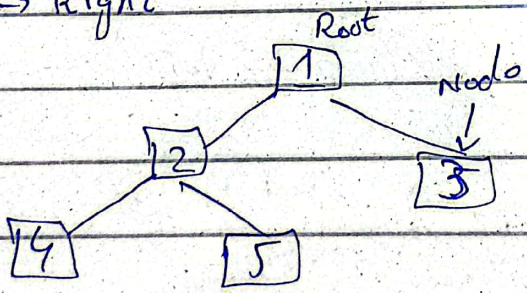


output = 4, 5

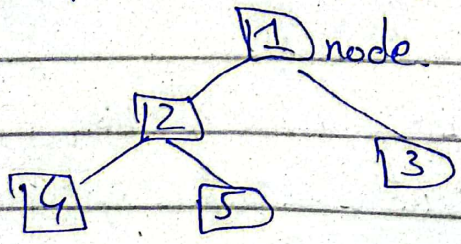


output = 4, 5, 2

Node → Right



output = 4, 5, 2, 3



output = 4, 5, 2, 3, 1