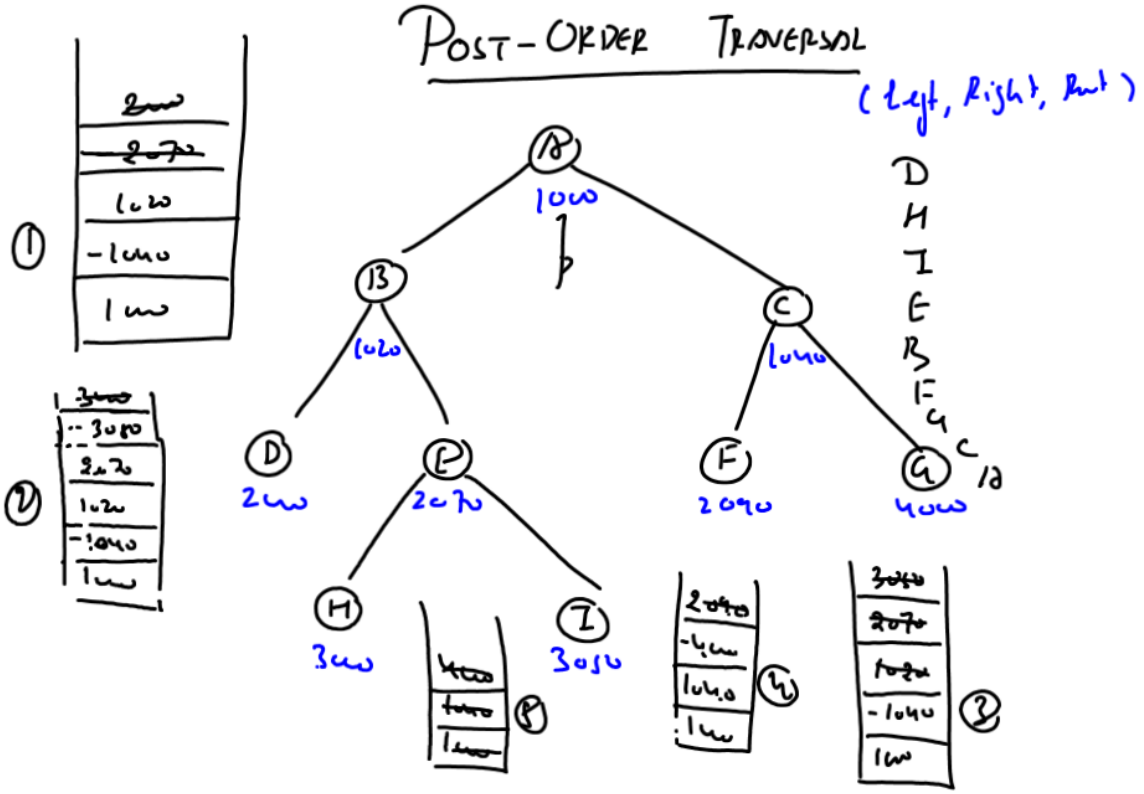
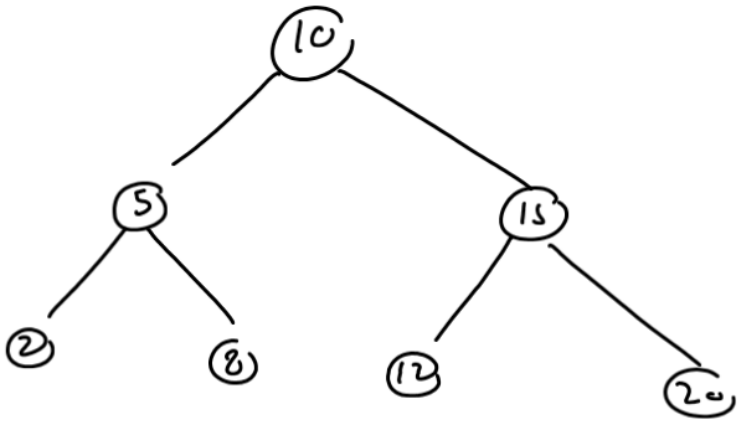
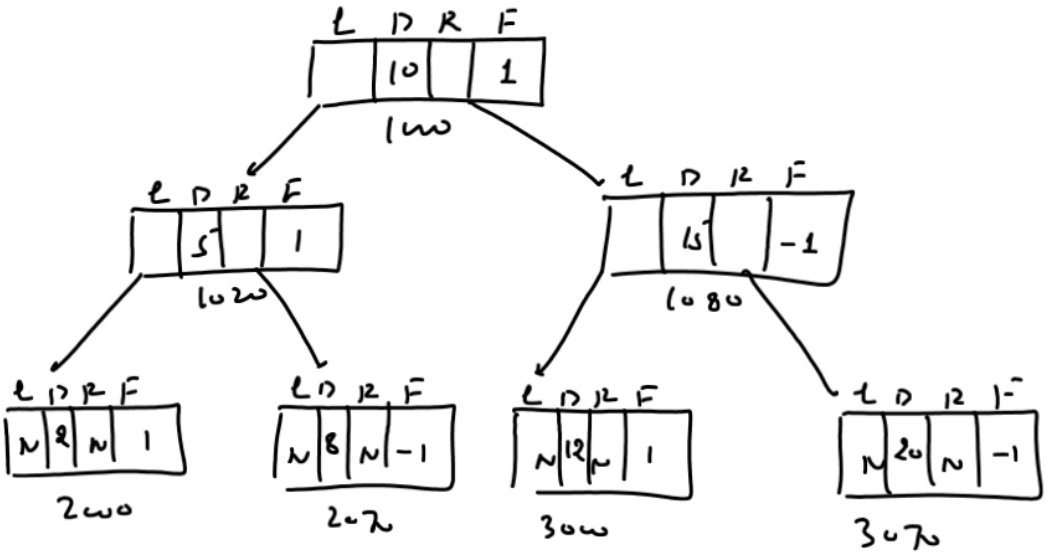


ALGORITHM FOR POSTORDER TRAVERSAL

1. Check whether the TREE is empty or not.
2. If it is EMPTY then print EMPTY TREE and rerturn.
3. Start from the CURRENT node and PUSH its ADDRESS in the STACK
4. If the CURRENT node has RIGHT child then PUSH its ADDRESS also as NEG in STACK
5. Move towards LEFT.
6. Repeat steps 3 to 5 until pointer becomes NULL.
7. POP the top node from STACK.
8. If we get NULL from STACK then finish and return.
9. Otherwise check the ADDRESS:

    a. If it is positive then print the DATA and goto step 7.
    b. If it is NEGATIVE then make it POSITIVE and goto step 3


10
5
15
2
8
12
20


L D R F
10 1
1000
L D R F
5 1
1020
L D R F
15 -1
1080
L D R F
N 2 N 1
2000
L D R F
N 8 N -1
2020
L D R F
N 12 N 1
3000
L D R F
N 20 N -1
3020

```
struct bst
{
    struct bst *right;
    int data;
    struct bst *left;
    int flag;
};
// Write further code
```

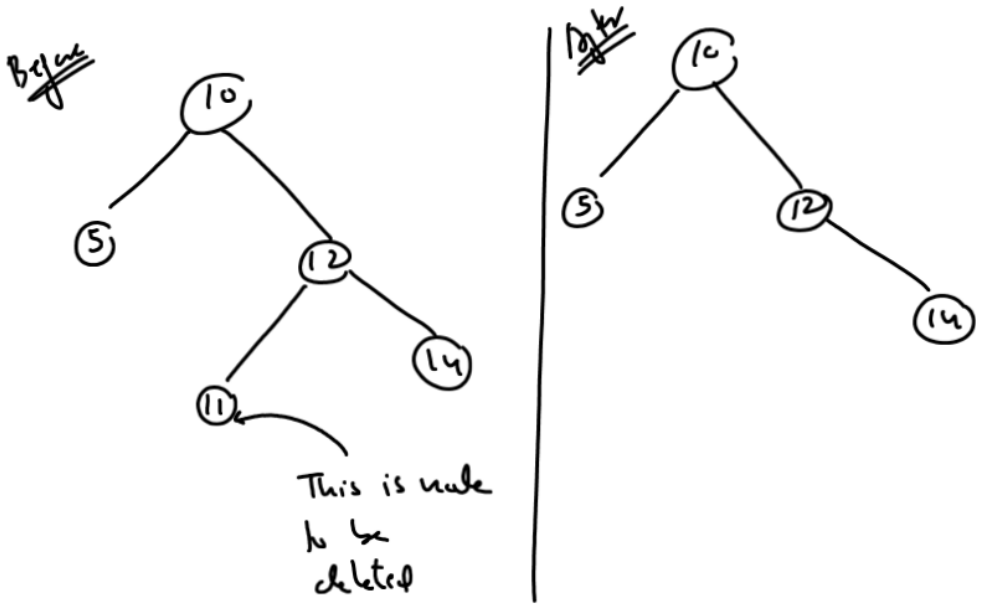
Deletion In BST

① Node to be deleted is leaf node

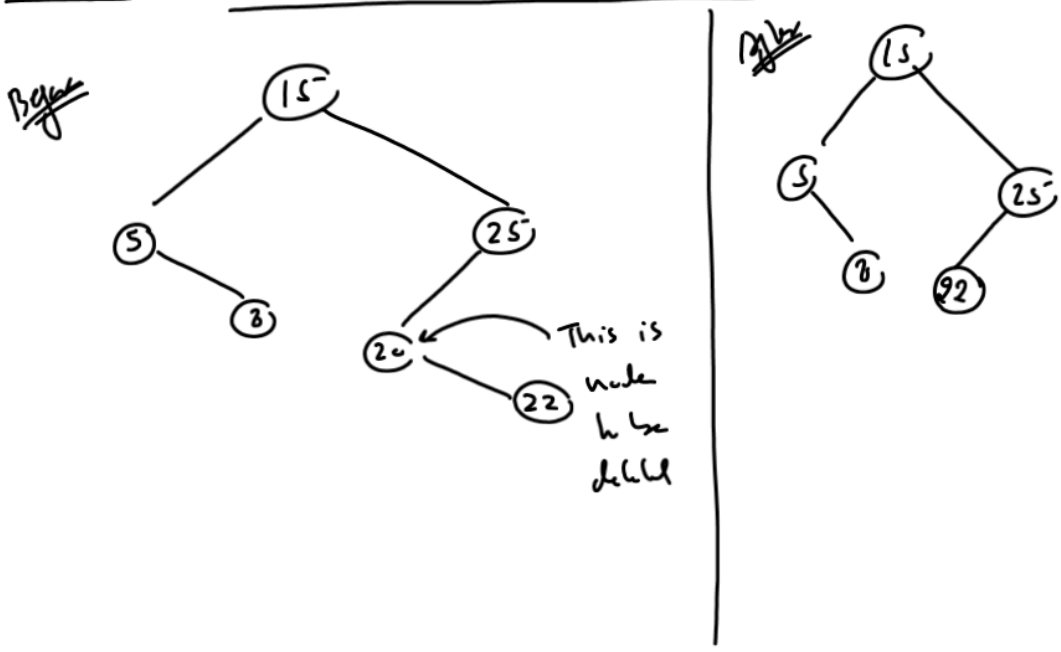
② " " " has just 1 child

③ " " " " 2 children

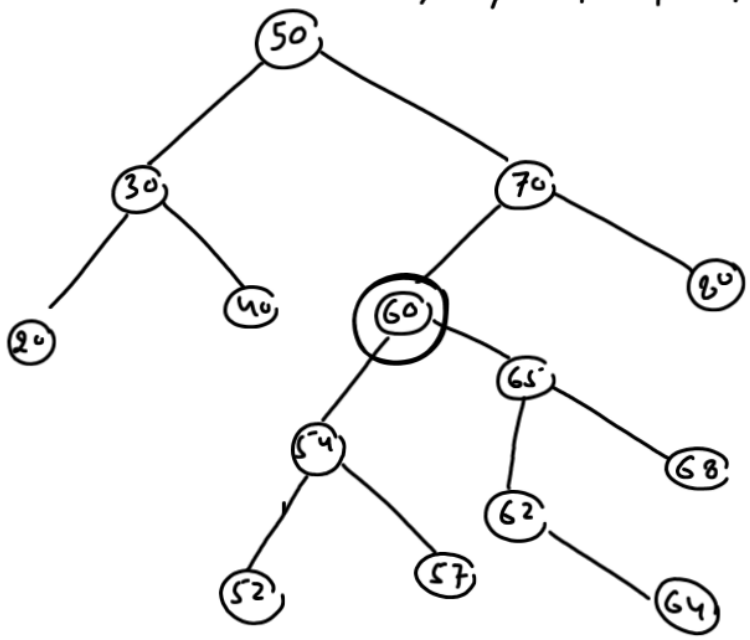
Case 1: Node To Be Deleted Is Leaf Node.

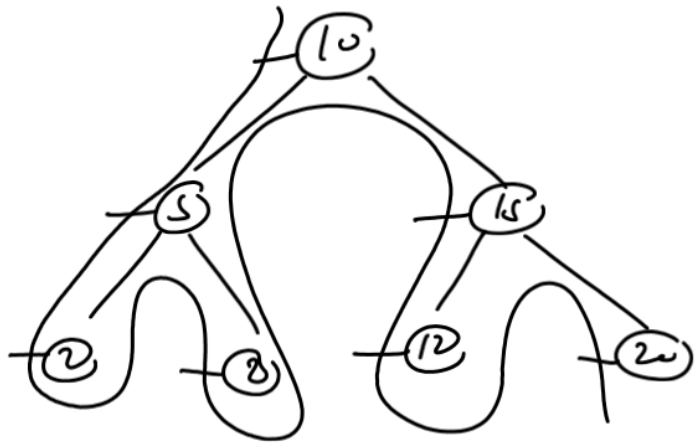


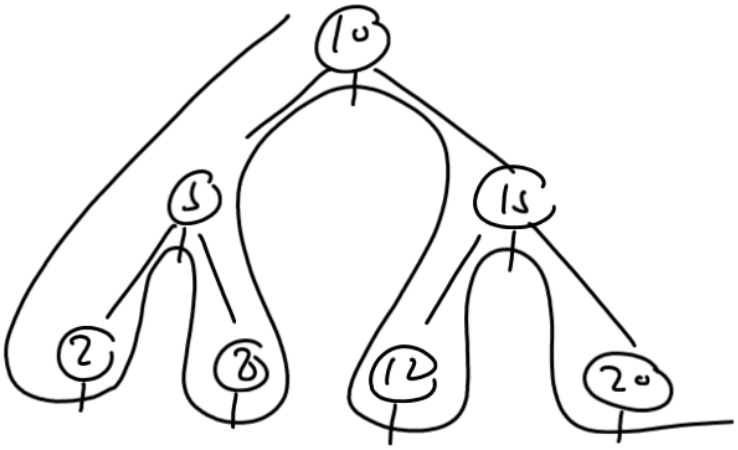
Case 2 : Node To Be Deleted Has 1 Child

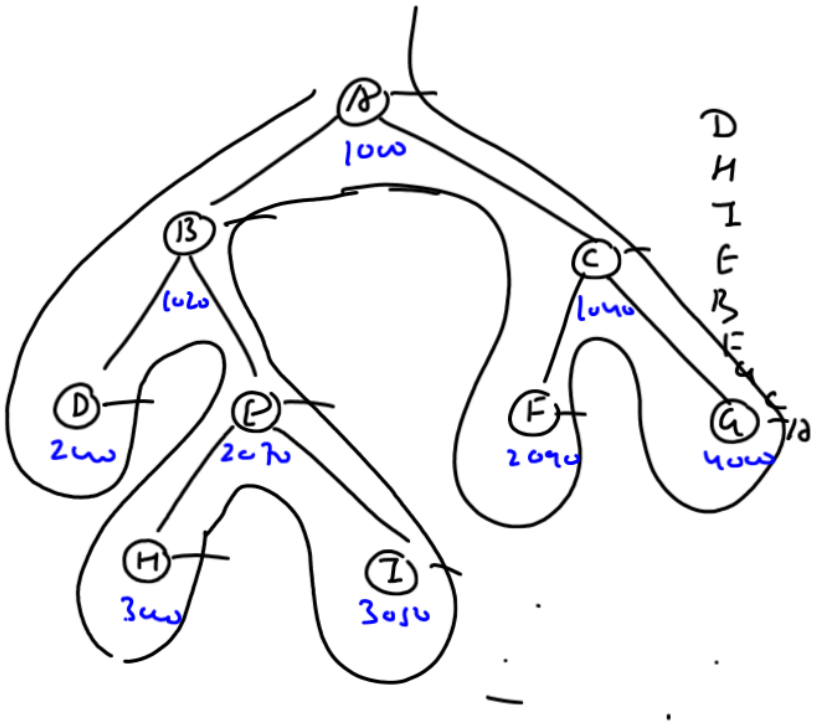


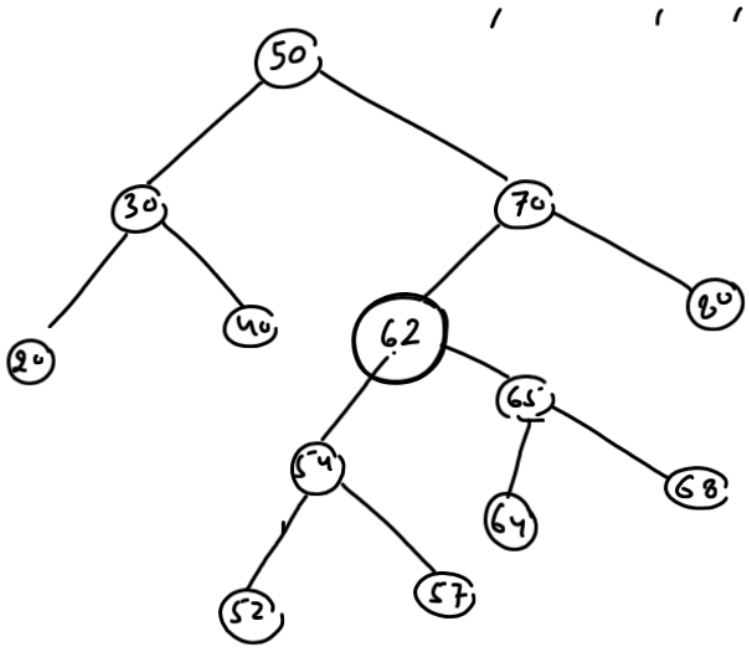
20, 30, 40, 50, 52, 54, 57, 60, 62, 64, 65, 68, 70, 80






10
5
15
2
8
12
20


A
1000
B
1020
C
1040
D
2000
E
2070
F
2090
G
4000
H
3000
I
3050


50
30
70
20
40
62
80
54
65
68
64
52
57