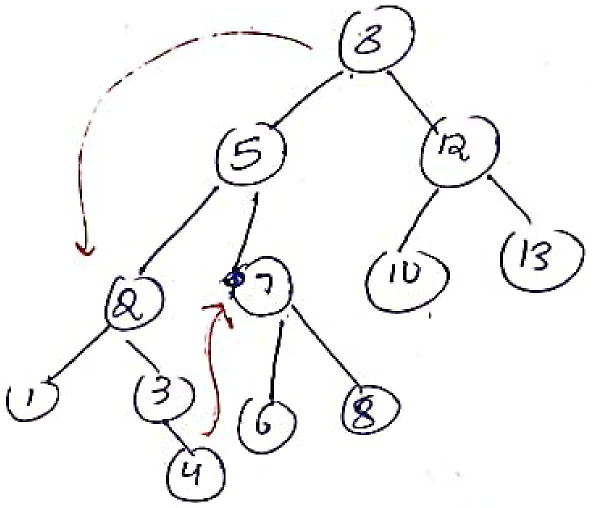
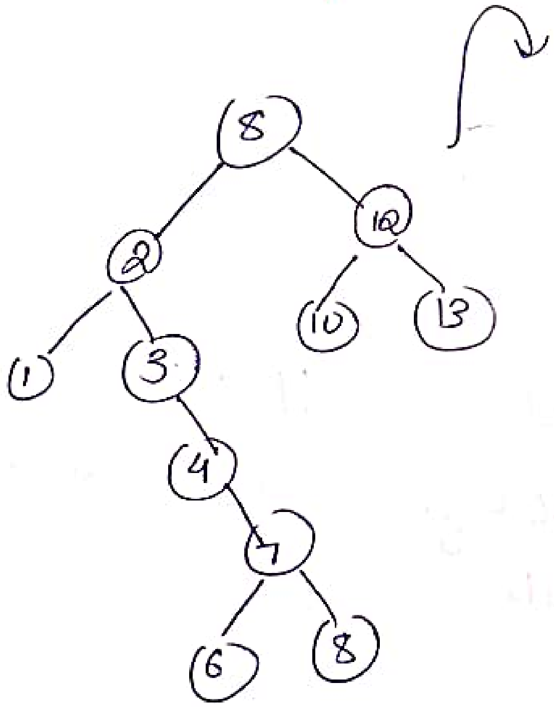
DELETION IN BST



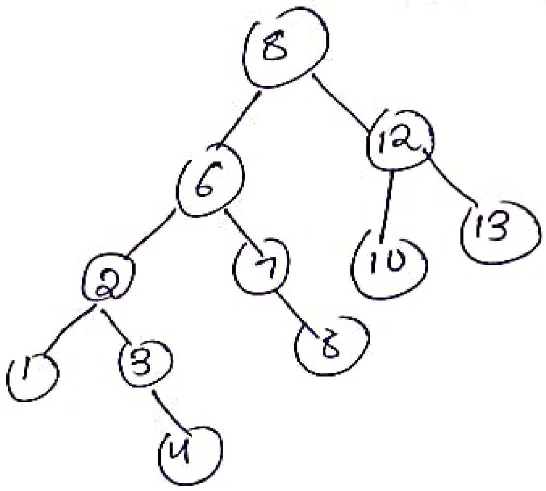
delete = 5

two ways

1#



2#



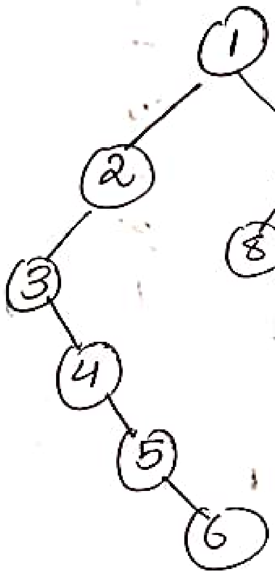
(1)

```
deleteNode(root, key){
        if(root = null) return null;
        if (root.val = key) return helper(root);
    Treenode dummy = root;
    while(root != null){
            if(root.val > Key){
                if(root.left != null && root.left.val == Key){
                        root.left = helper(root.left);
                        break;
                }
                else root = root.left;

        }else{
            if (root.right != null && root.right.val == key){
                    root.right = helper(root.right);
                    break;
            }
            else root = root.right;
        }
    return dummy;

}
```

```
public TreeNode helper (root) {
        if (root.left == null) return root.right;
        else if (root.right == null) {
                        return root.left;

            }

        else {
            Tree. rightchild = root.right;
                lastRight = findlastRight (root.left);
                lastRight.right = rightchild;
              return root.left;

          }

  }

findlastRight (root) {
            if (root.right == null) return root;
          return findlastRight (root.right)

        }
```

POSTORDER USING 1 STACK

left right root



either we have current node or there is something in stack to process

```
while(curr != null || !stack.empty()){

if(curr != null){
    st.push(curr)
    curr = curr -> left;
}
```

moving to left trying to find extreme left

when current becomes null

```
else
temp = st.top() -> right;
if(temp == null){
    temp = st.top()
    st.pop()
    post.add(temp);
while(!st.isempty && temp == st.top() -> right){
    temp = st.top(), st.pop()
    post.add(temp -> data);
}
else curr = temp;
```

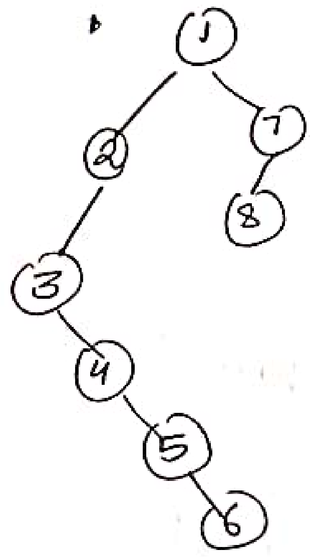
as right does not exist

then node will be checked if it is extreme left or right or none

→ either extreme left or right we need to print that

(last to read)

it checks if the node is extreme right then we process and pop the root also if extreme left then the above root node right will be processed



temp = 4 ~~5~~ ~~6~~ null
~~6~~ ~~5~~ ~~4~~ ~~3~~
null 2 null null

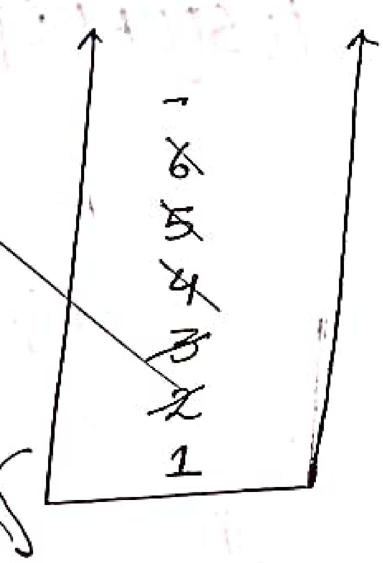
cur = ~~1~~ ~~2~~ 3 null

~~4~~ null

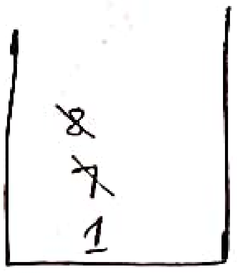
~~5~~ null

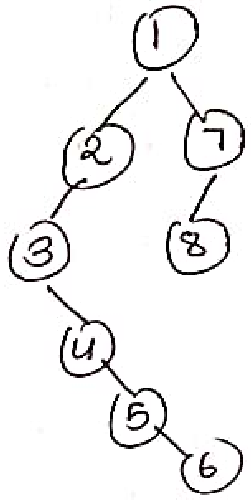
~~6~~ null

~~7~~ ~~8~~ null

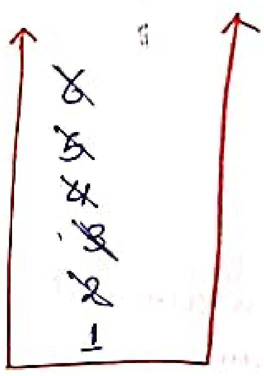


post → 6 5 4 3 ~~2~~ 8 7





temp = ~~3~~ ~~4~~ ~~5~~ ~~6~~ null
~~6~~ ~~5~~ ~~4~~ ~~3~~ null
~~2~~ ~~7~~ null
~~8~~ null
~~7~~ 1



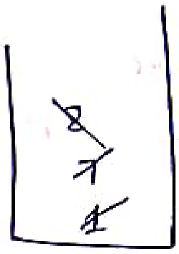
cur = ~~1~~ ~~2~~ ~~3~~ null

~~4~~ null

~~5~~ null

~~6~~ null

~~7~~ ~~8~~ null



post = 6 5 4 3 2 ~~7~~ 8 7 1