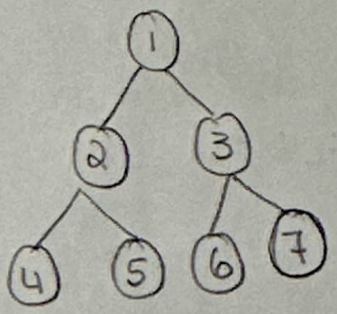
→ Flatten Binary Tree to LL:

① Using Rewrsion :

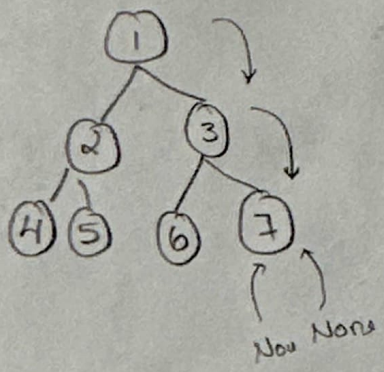


Idea

→ Go Right

→ Go Left

When returing keep track of the Node

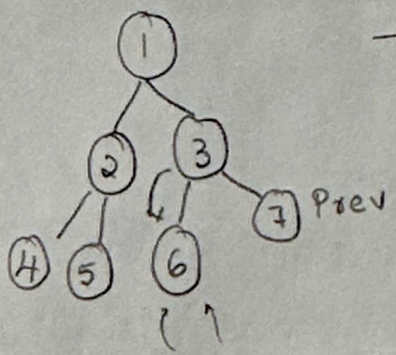


→ Node 7

Right = None

Left = None

PrevNode = 7



→ Node 6

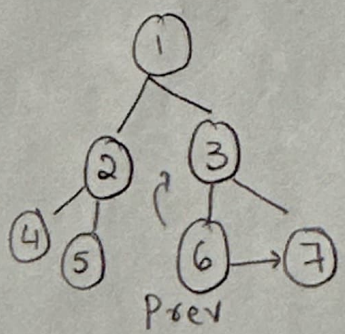
Right = None

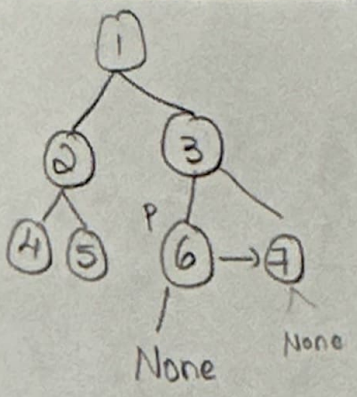
Left = None

Since prevNode = 7

Make Node 6 → right Node as 7

return 6 as prev Node



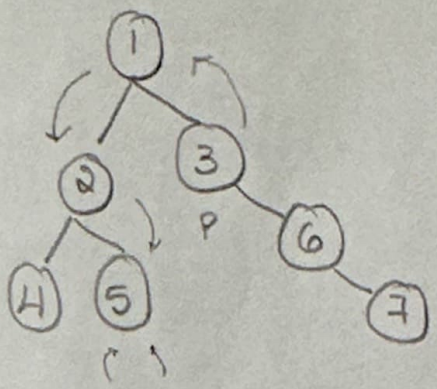


→ Node 3

Since PrevNode = 6

Node3 .right = 6

return Node 3 as prevNode



→ Node 2 :

→ Go Right

→ Go Left

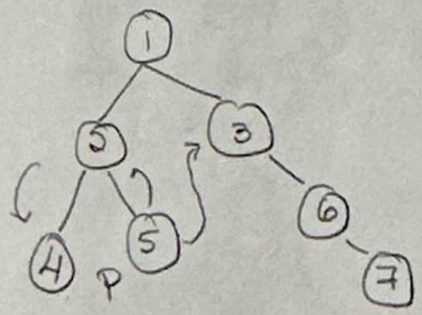
→ Node 5 :

Right = None

left = None

Make node 5.right = Node 3

return node 5 as prevNode



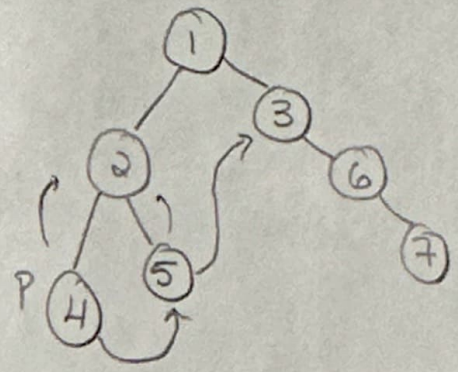
→ Node 4

Right = None

Left = None

node 4. right = node 5

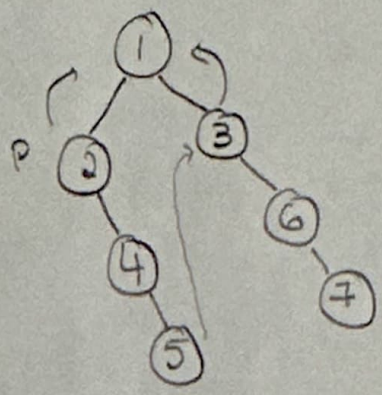
return node 4 as prevNode



→ Node 2

node 2. right = node 4

return Node 2 as prevNode.

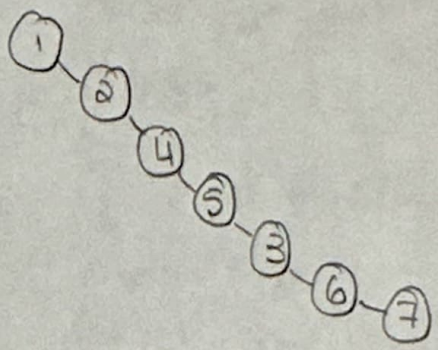


→ Node 1

node1 . right = node 2

and Prev = node 1

since this is the root we return



Idea: Moving from Right to Lyt, Make the Lyt child point to right child.

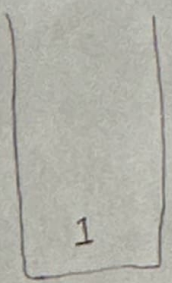
- Break Connection b/n Parent & Right child
- Make the Left child as the Right child of parent

---

② Using Iteration

same Idea: Move from Right to Lyt

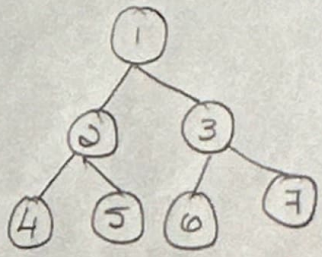
* Make the parents Left child as parents right child
* Make the Right child as Left childs (right most) right child



→ move from right to Lyt

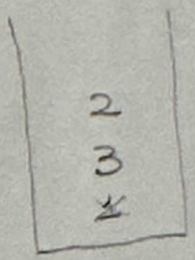
node.right = 3

node.Lyt = 2

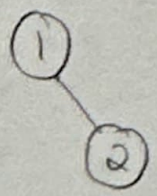


Node 1

Right child = 3
Lyt child = 2



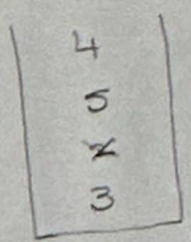
node 1 . right = top of stack (Lyt child)



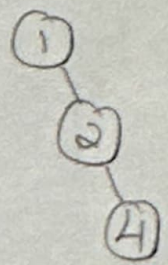
Node 2

Right child = 5

Lyt child = 4



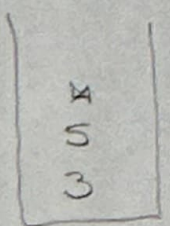
node 2 . right = top of stack



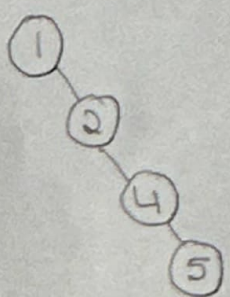
Node 4

Right : None

Lyt = None



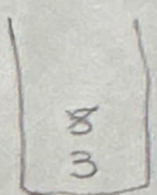
node 4 . right = top of stack (this will be the right child)



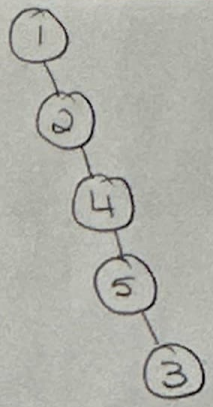
Node 5

Right = None

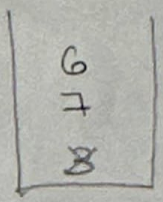
Lyt = None



node 5 . right = top of stack (this will be right child)



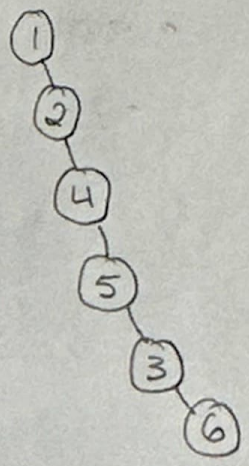
Node 3



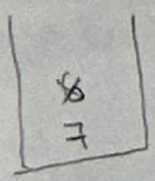
Right = 7

Left = 6

node.right = 6



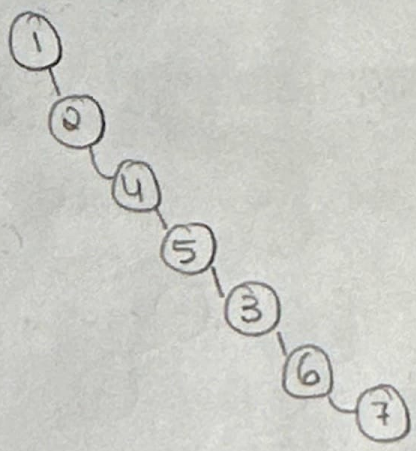
Node 6



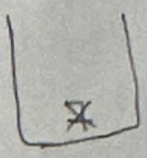
Right = None

Left = None

node.right = 7



Node 7



Right = None

Left = None

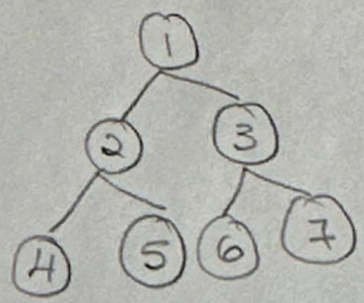
Since there are no more Nodes Left, we return

Approach 3 : Threaded Tree

The idea we were using till now was to take the leftMost tree's (right most child) and connect it to right child

This is just what threaded Tree is

Eg:



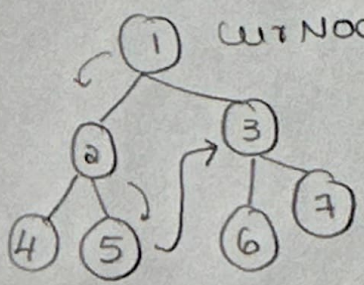
2 step

(1) connect left childs right most Node to right child

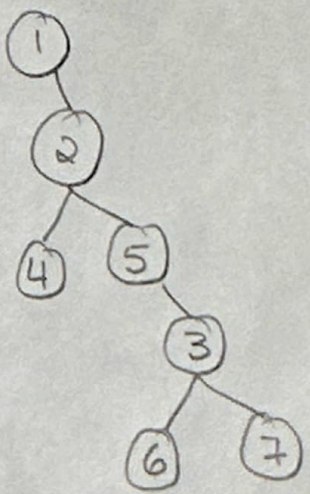
(2) Make the Left child as the right child.

→ Node 1

i)

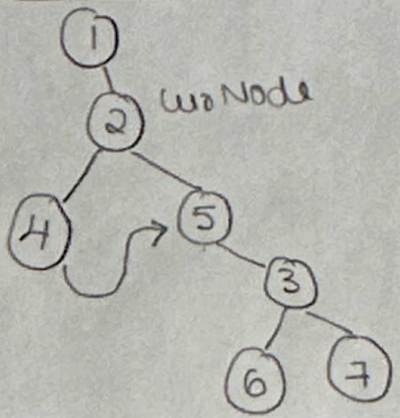


ii)

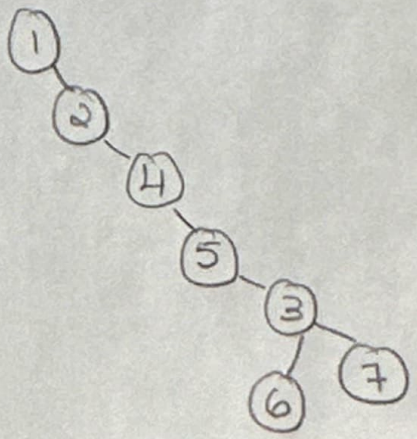


→ Node 2

i)

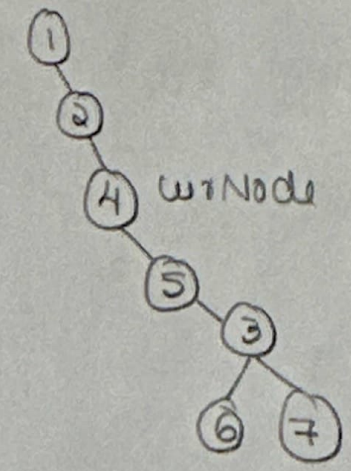


ii)



→ Node 4

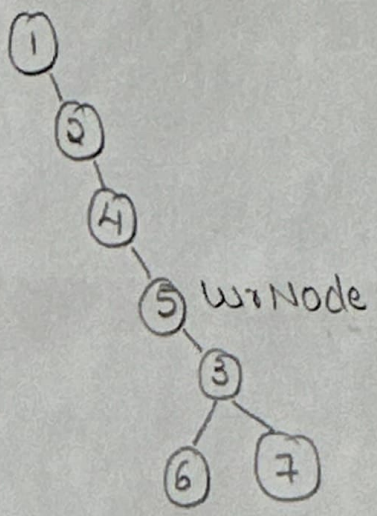
i)



Since there is no Lyt child, we just move forward

curNode = curNode.right

→ Node 5

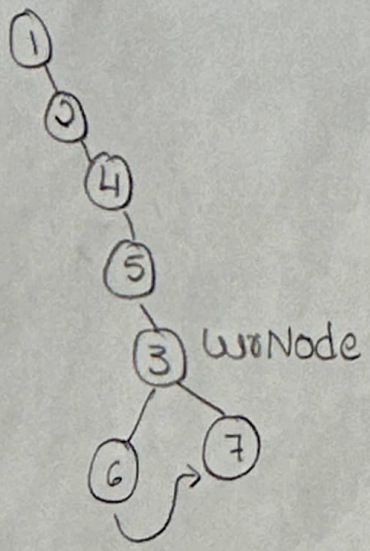


There is no Lyt child

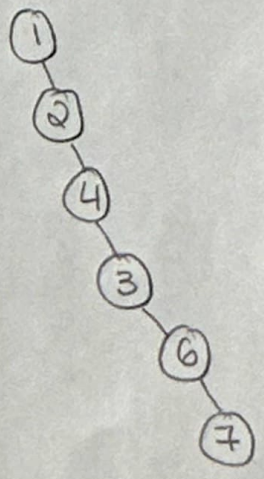
curNode = curNode.right

→ Node 3

i)



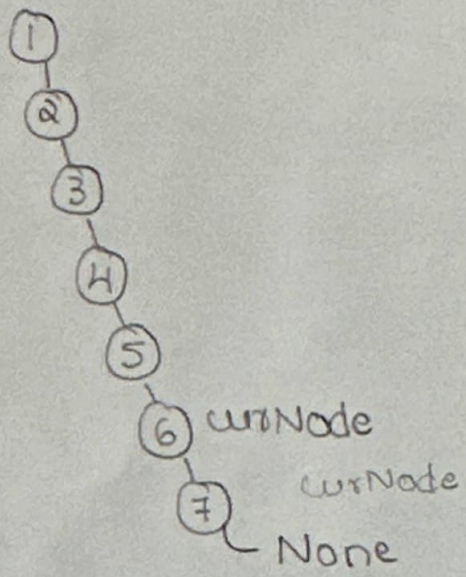
ii)



→ Node 6 & Node 7

No left child

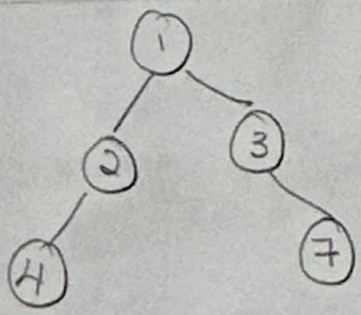
currNode = currNode.right



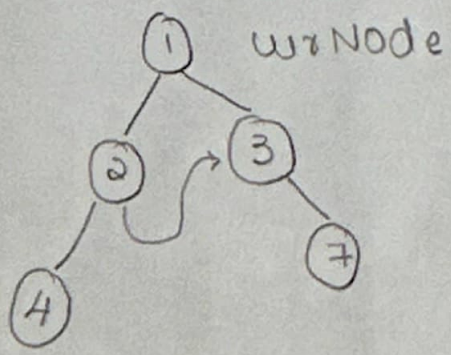
→ Node None

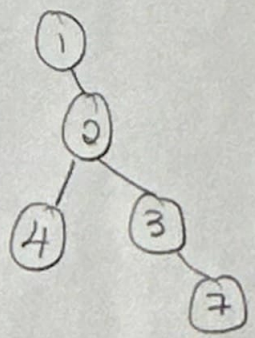
Since we reach the end, we return

Eg

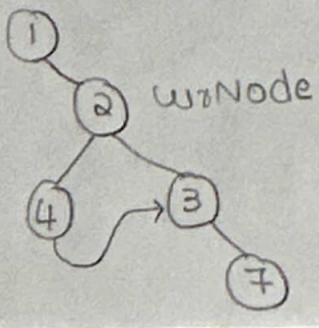


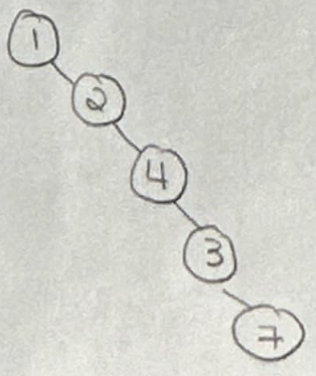
→ Node 1



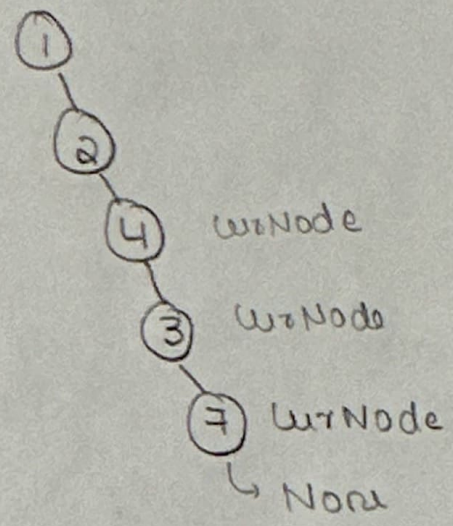


→ Node 2





Node 4, 3, 7:



Since there is No Left Node
We move right

currNode = currNode.right

→ None

We reached the End, we return.