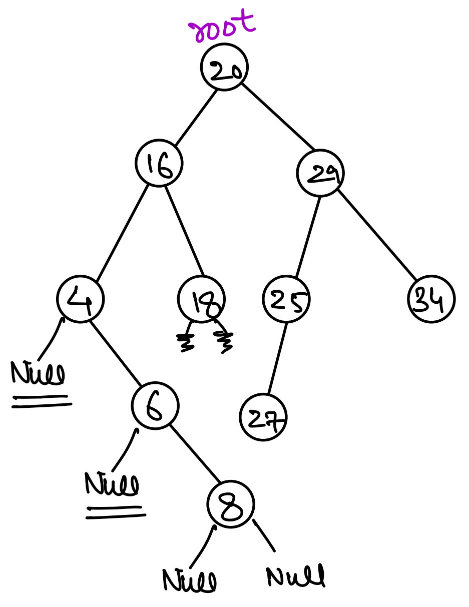
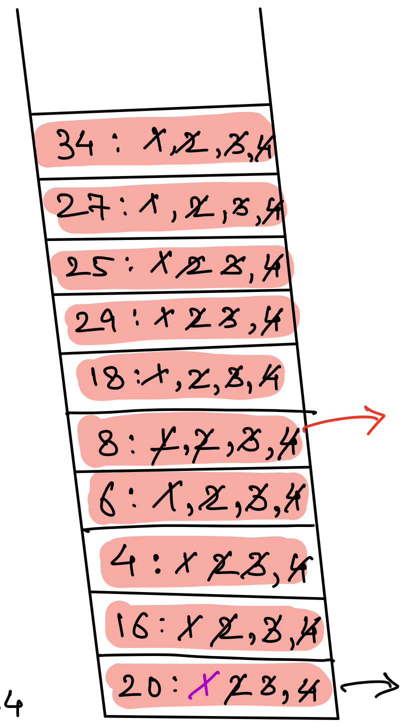
Inorder :-

Left Root Right

```
void inOrder (root) {
    1) if (root == Null)
              return;
    2) inOrder (root.left);
    3) print (root.data)
    4) inOrder (root.right);
}
```



4, 6, 8, 16, 18, 20, 27, 25, 29, 34



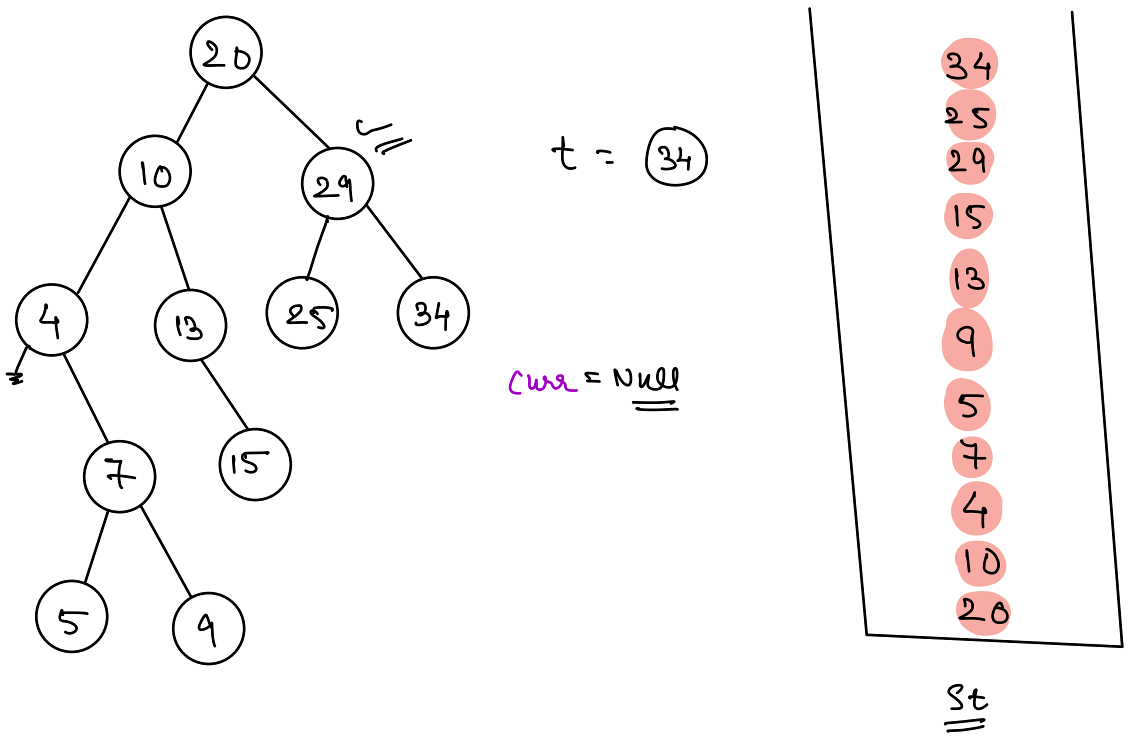
Call Stack

Idea :-

1) Till you get a Null on the left side, keep pushing into the stack.

2) If root == Null, get the top element from stack & go to right.

```
void   inOrder (root) {
       stack < Node > st ;
       Node  curr = root ;
        while ( curr != Null || st.size()>0)  {

              if ( curr != Null ) {
                     st.push ( curr ) ;
                     curr = curr. left ;
              }
              else {
                     Node temp = st. top () ;
                     st. pop () ;
                     print ( temp.data ) ;
                     curr = temp. right ;
              }
        }
}
```



4, 5, 7, 9, 10, 13, 15, 20, 25, 29, 34

TC : O(N)

SC : O(H)

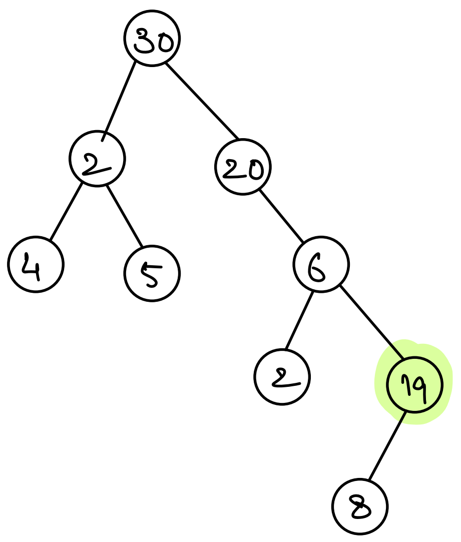
↳ WC : O(N)

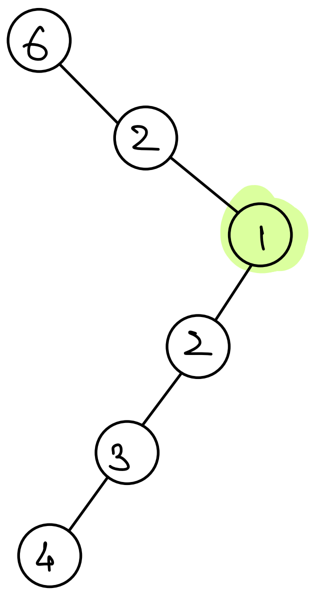
HW

1) Preorder Iterative

2) Postorder Iterative.

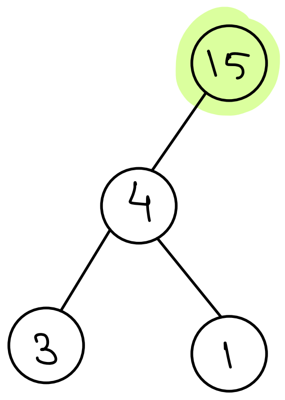
Q: Given a tree, find the last inorder node that gets printed.



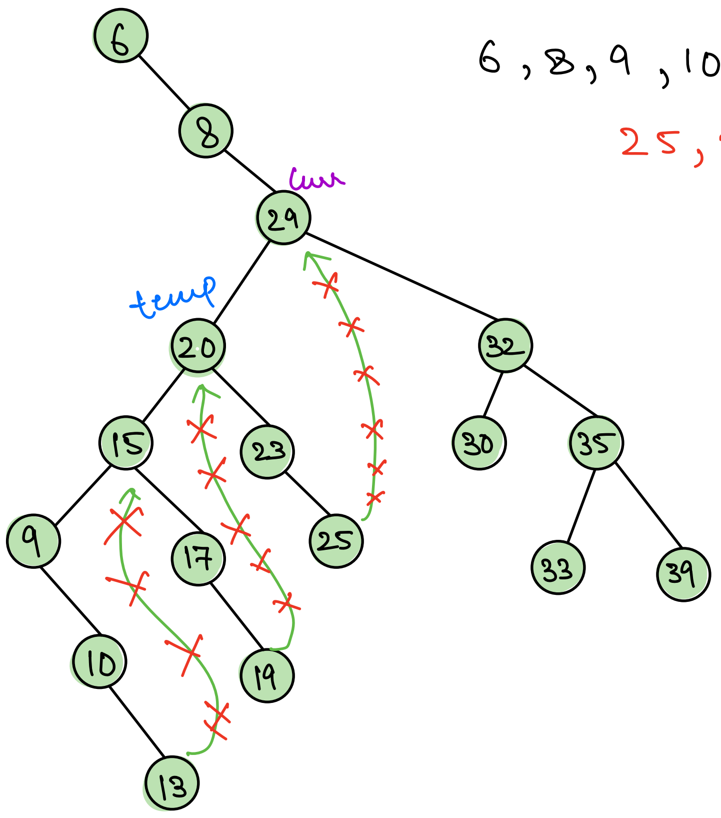


⇒ Keep going on Right side until we get Null.

⇒ Right most node.



Q: Can we do Inorder traversal in O(1) SC.



6, 8, 9, 10, 13, 15, 17, 19, 20, 23
25, 29

⇒ If left child is NULL, print the curr node & go to right side.

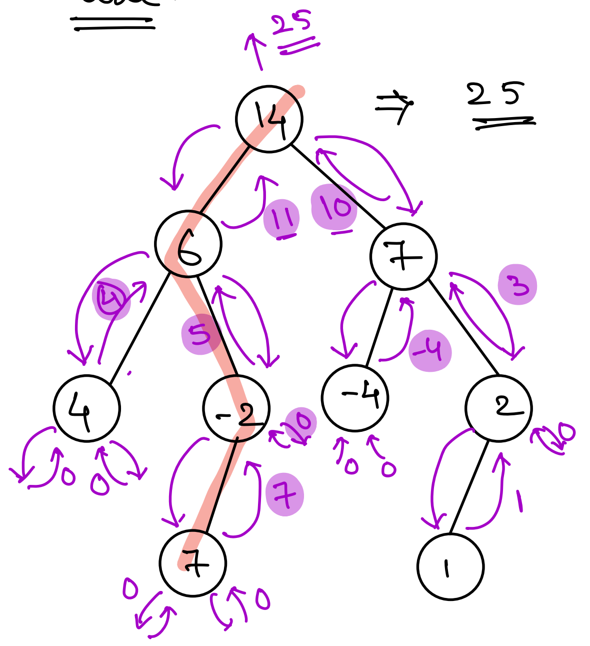
11:05

```
void    inOrder ( root ) {
      Node  curr = root ;
      while ( curr != Null ) {

          if ( curr.left == NULL ) {
                  print ( curr.data ) ;
          }       curr = curr.right ;
          else {
              Node  temp = curr.left ;
              while ( temp.right != Null &&
                              temp.right != curr ) {

                   temp = temp.right ;
              }
              if ( temp.right == Null ) {
                     // Visiting curr node for 1st time.
                     temp.right = curr ;
                     curr = curr.left ;
              }
              else { // Visiting curr Node for 2nd time
                     temp.right = Null
                     print ( curr.data ) ;
                     curr = curr.right ;
              }

      }
```

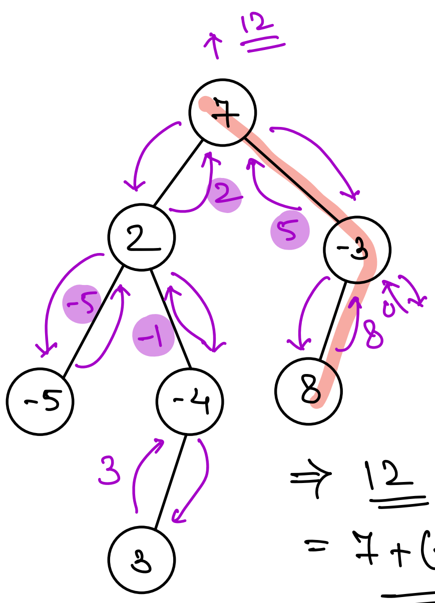
⇒ Morris Inorder Traversal.

$$\boxed{\begin{array}{l} TC :\ O(N) \\ SC :\ O(1) \end{array}}$$

Q: Find the max sum path starting from root node.

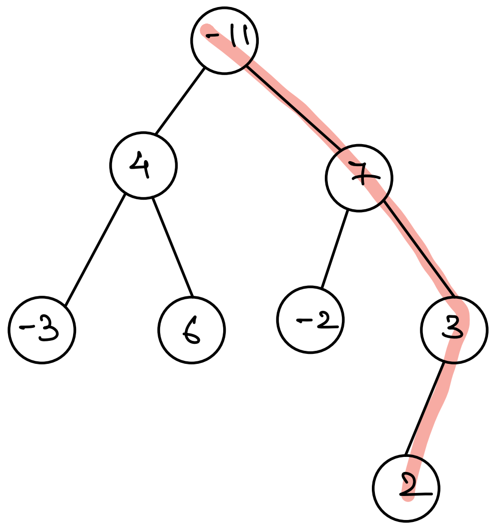


$\Rightarrow$ $\underline{25}$

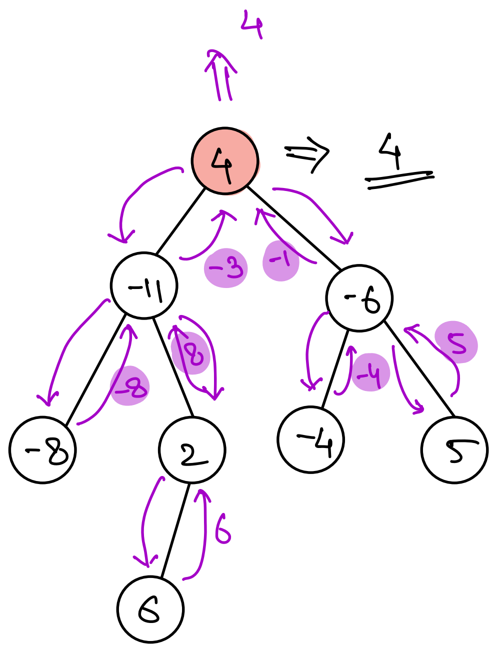


$\Rightarrow$ $\underline{12}$

$= \underline{7+(-3)+8}$



$\Rightarrow 1$



$\Rightarrow$ $\underline{4}$

$\Rightarrow$ $\underline{-10}$.

Single node can also be an ans.

```
int   maxPathSum ( root) {
      if ( root == Null)
            return 0;
      int l = maxPathSum(root.left);
      int r = maxPathSum(root.right);
      return root.data + max(l,r);
}
```

$\underbrace{\text{max}(l,r)}$

↓

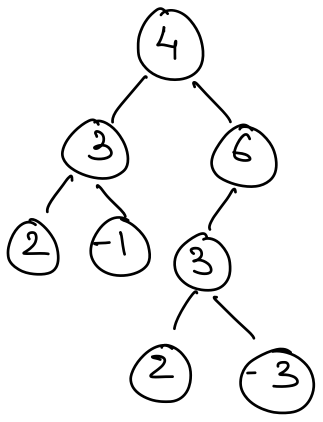
max (l , r , 0)

max(max(l,r), 0)

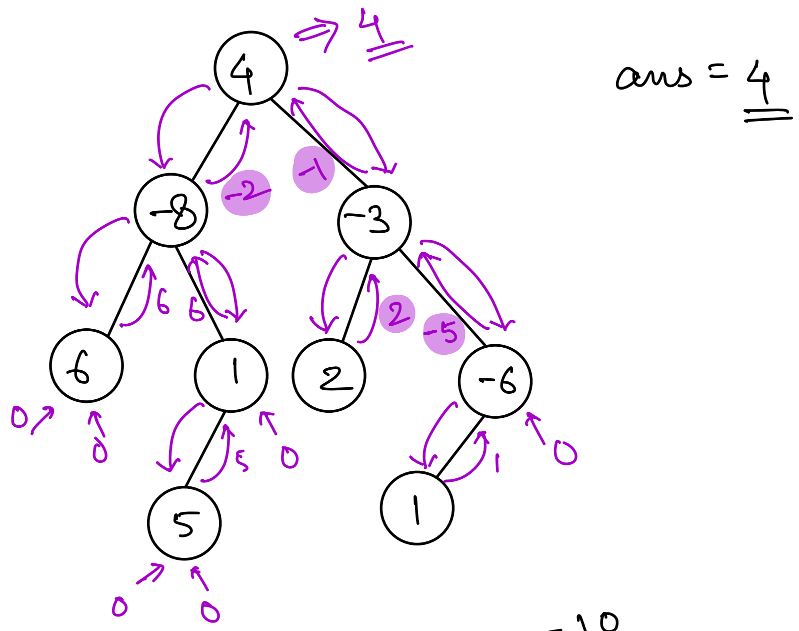
TC : O(N)

SC : O(N)

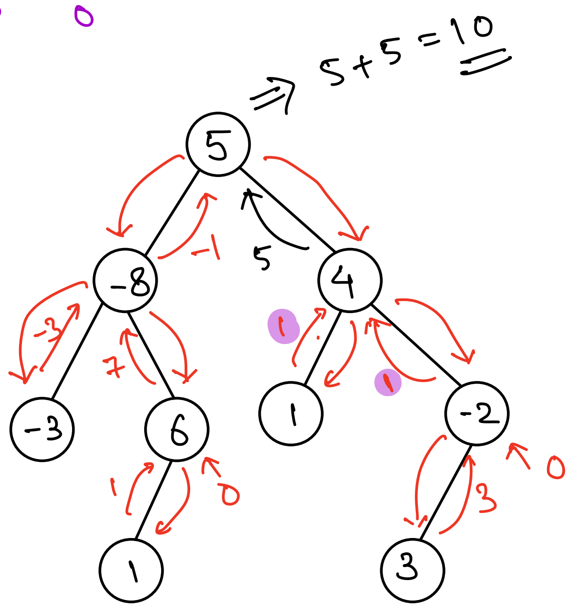
Max Path Sum containing root node.



ans = 20.



⇒ 4
ans = 4
4
-8
-3
6
1
2
-6
5
1
-2
-1
6
6
2
-5
5
1
0
0
0
0
0
0



⇒ 5+5 = 10
5
-8
4
-3
6
1
-2
1
3
-1
5
-3
7
1
1
1
0
3
0