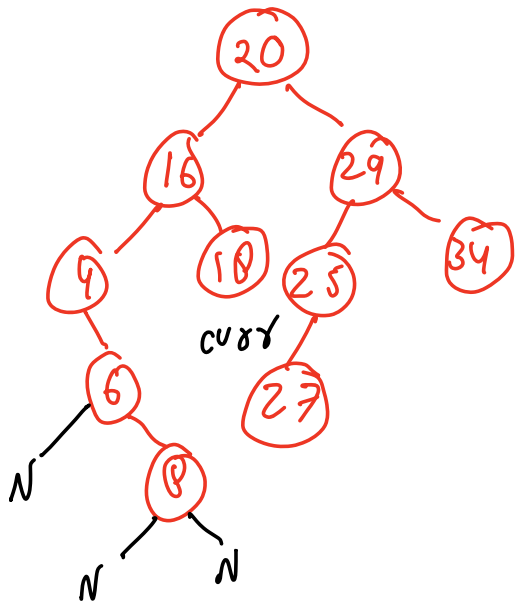
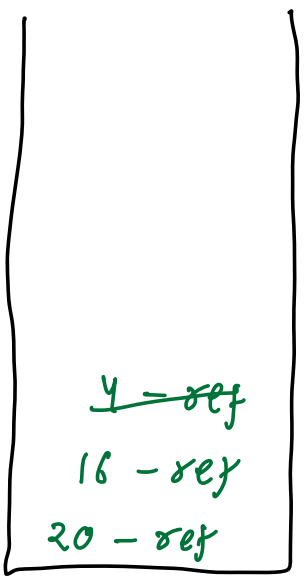
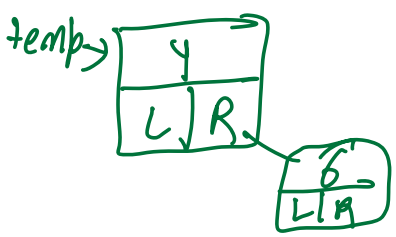
Inorder

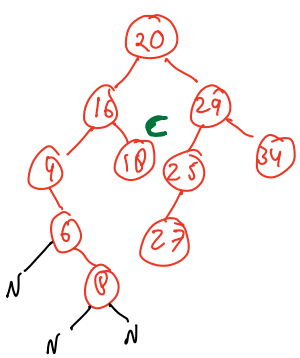
Left root right





4, 6, 8, 16



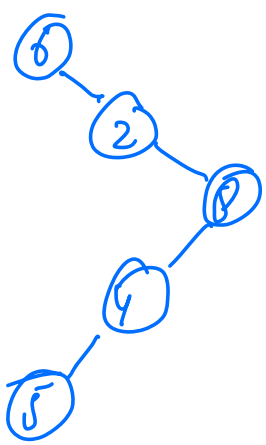



```
void inorder (Node root)
{
    stack < Node > s;
    Node curr = root;
    while ( curr != Null || s.size() > 0)
    {
        if (curr != null)
        {
            s.push( curr)
```

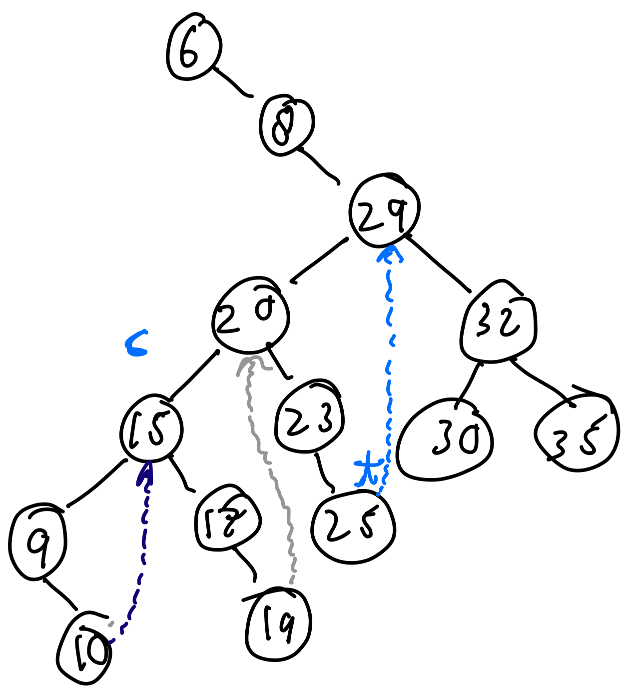
4, 6, 8, 16

```
            curr = cur.left;
        }
        else
        {
            Node temp = s.top()
            s.pop()
            print( temp.data);
            curr = temp.right;
        }
    }
}
```

Q2⇒ Last value of inorder



Morris Inorder Traversal



6, 8

9, 10

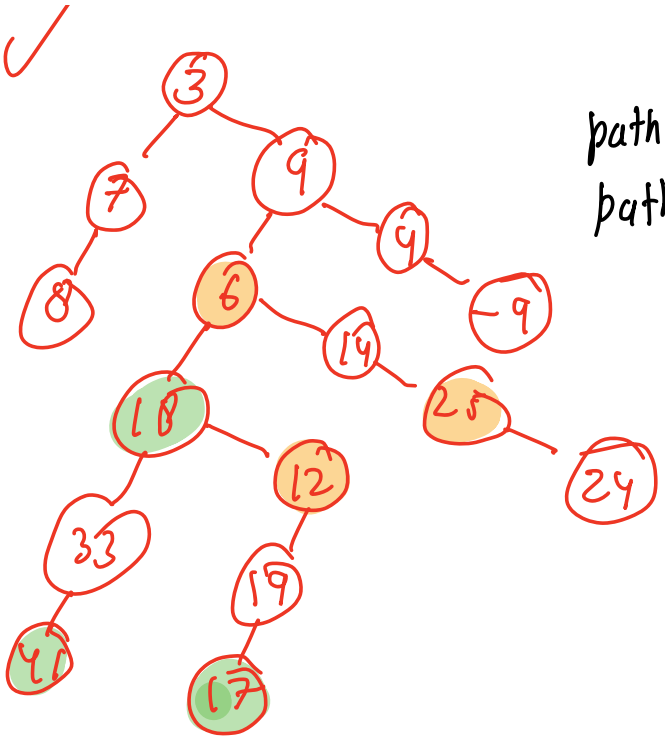
```
void moris Traversal (Node root)
{
    Node curr = root;        while (curr != null)
                             {
     if ( curr.left == Null )       O(N)
      {
        print (curr.data)
        curr = curr . right;
      }
     else
     {
      Node temp = curr.left
```

```
while (temp.right != null && t.right != cur)
    { temp = temp.right }
if (temp.right == null)
 { temp.right = cur;
    cur = cur.left;
 }
 if (temp.right == cur)
 {
   temp.right = null;
    print(cur.data)
    cur = cur.right;
 }
}
)
```

Break: 9:03 AM

Q) LCA

Given 2 value in BST, find shortest path between them.



path(41) = 3, 9, 6, 18, 33, 41

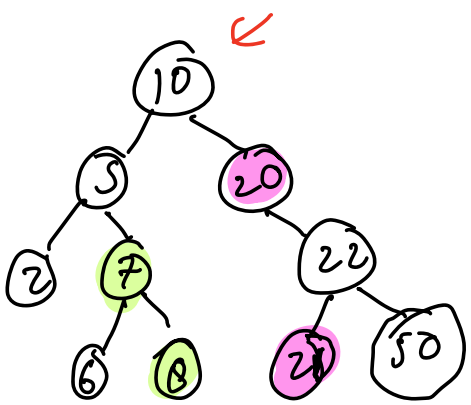
path(17) = 3, 9, 6, 18, 12, 19, 17

path (a) from root

path (b) from root

stop at last comon node blw thm

traverse array to get an



TC: O(H)

SC: O(H)

```
int LCA in BST (Node root, n1, n2)
{
    if (root == null)    return 0;

    if (root.val > n1 && root.val > n2)
        ret LCA in BST (root.left, n1, n2)
    else if (root.val < n1 && root.val < n2)
        ret LCA in BST (r.right, n1, n2)
    else
        ret root.val;
}
```

```
void  moris Traversal  (Node  root)
  {
     Node   curr = root;  ←  while (curr != null
                              {
     if ( curr.left  == Null )          O(N)
        {
          print   (curr.data)
          curr = curr . right;
        }
     else
        {
          Node temp = curr. left
  while  (temp. right != null   && t.right != curr)
          {   temp =   temp. right   }
     if ( temp. right == null)
       {  temp. right  =  curr;
          cur  = cur . left;
       }
       if (temp . right ==  curr)
       {
         temp . right  = null;
         print (  cur . data)
          cur  =  cur . right ,
       }
    }
  )
```