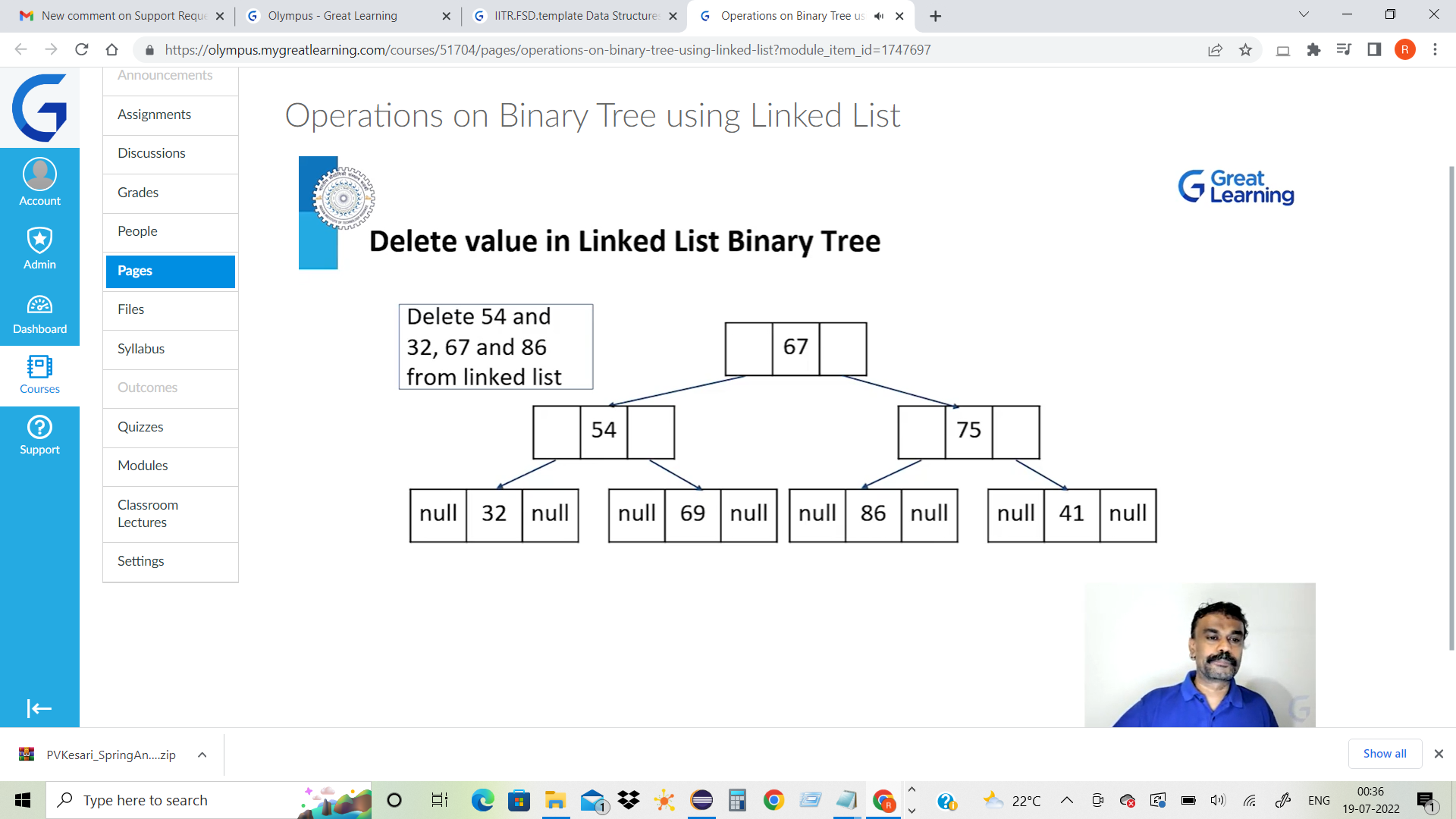
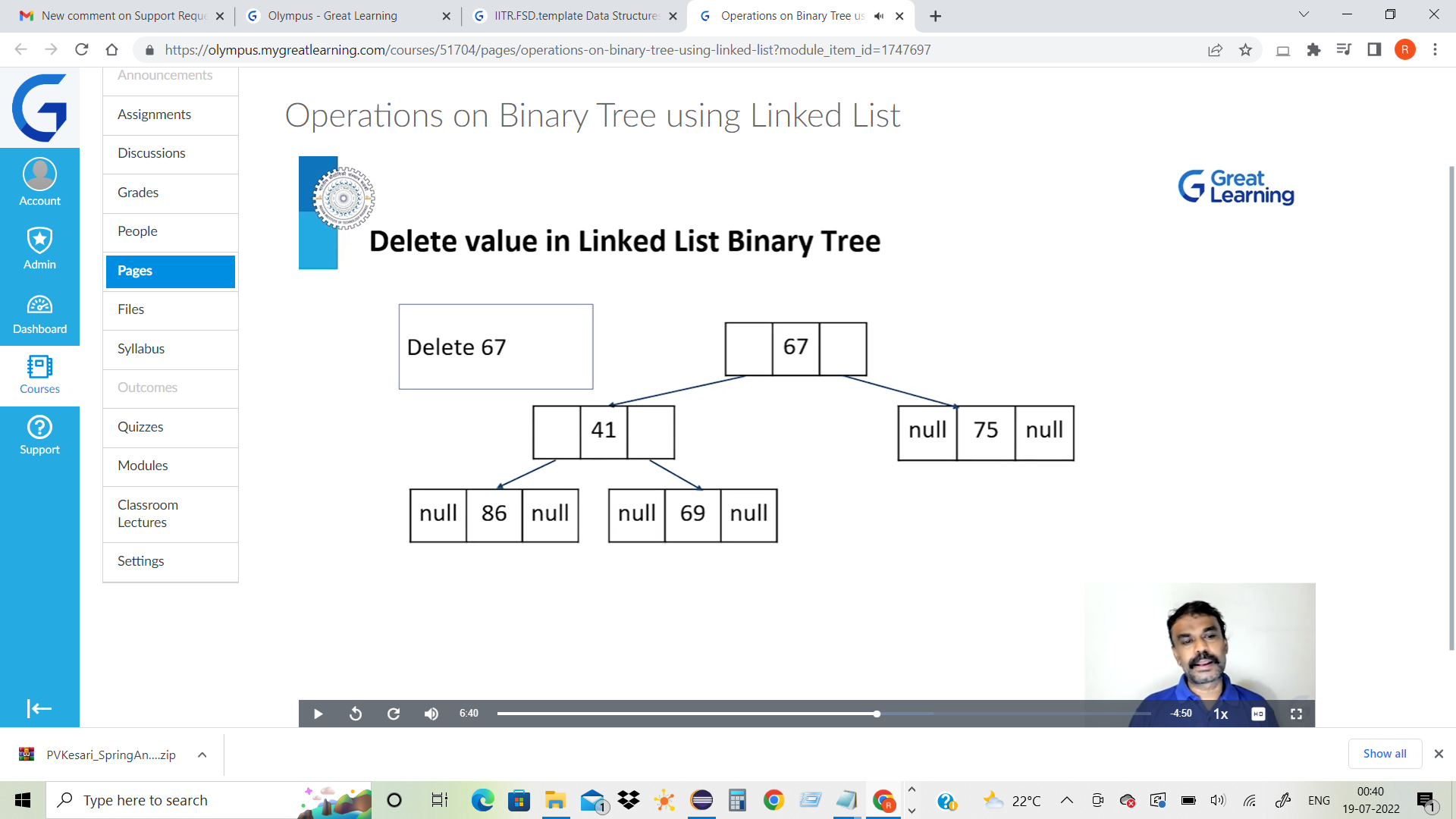
Condition:

Find the deepest rightmost node, replace it with the deletion node and delete the deepest rightmost node

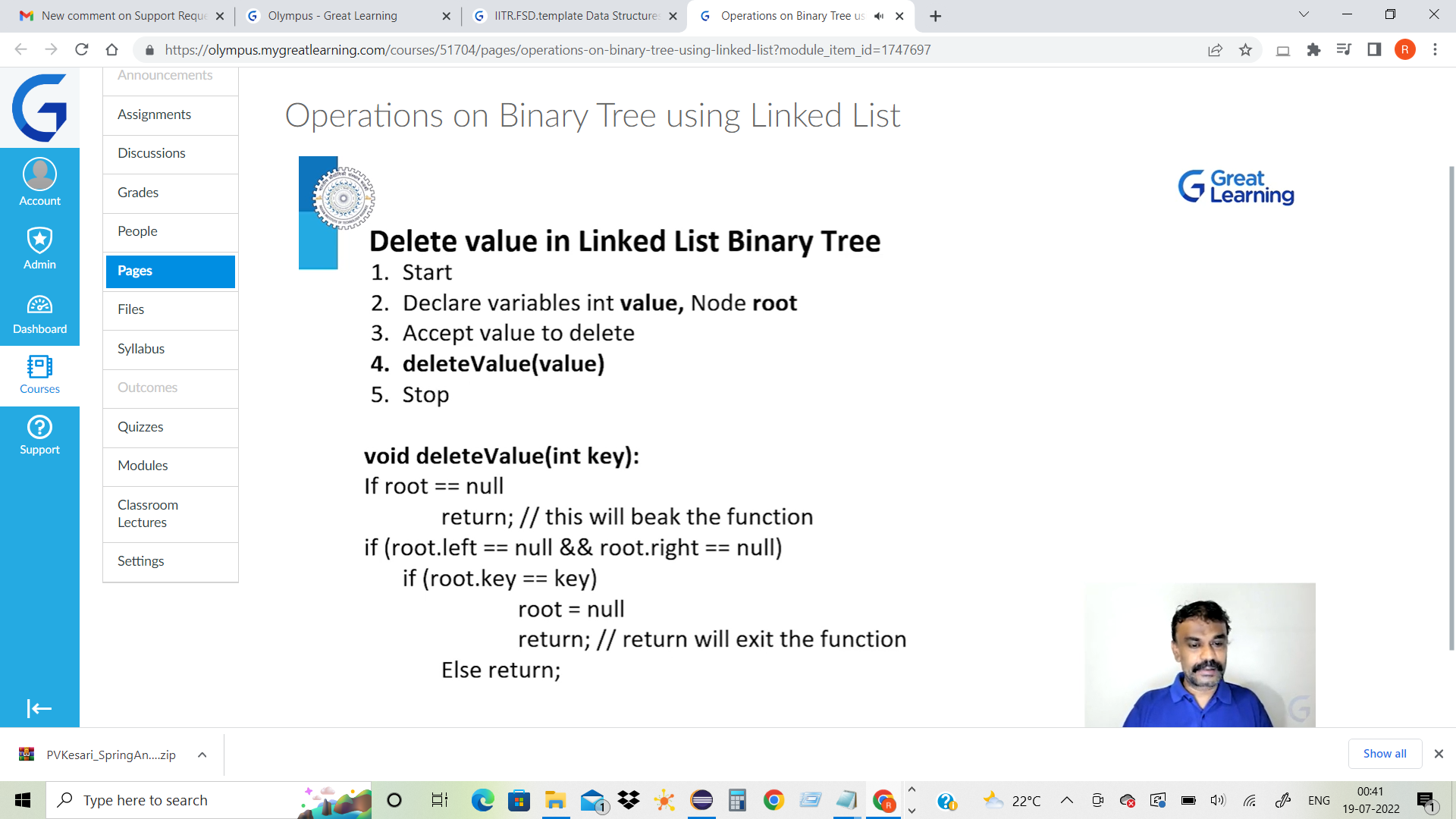


If for eg. 54 is to be deleted in the above tree, the right most node is to be found ie 41 in this case

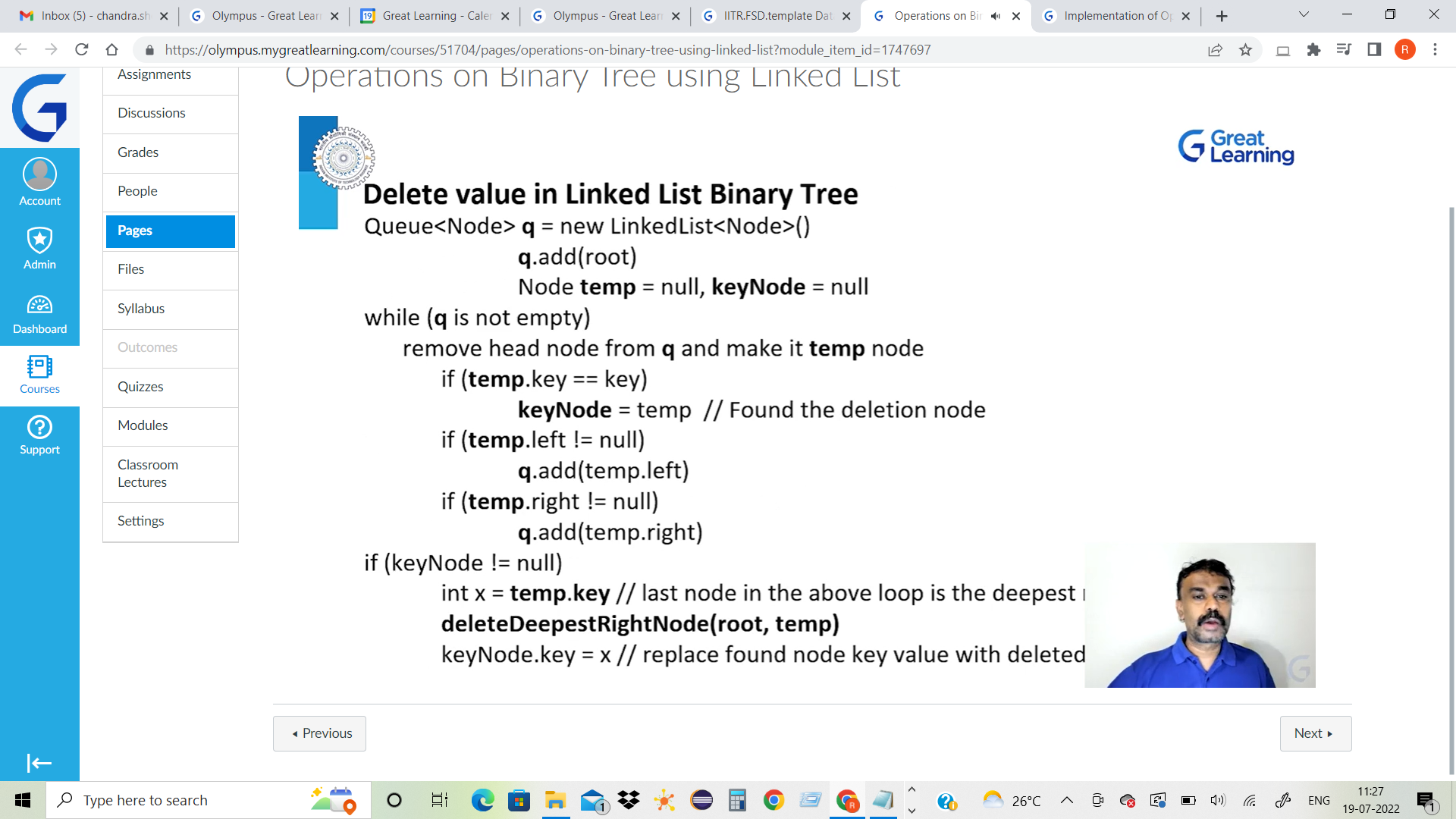
Replace 54 with 41 and delete the deepest right most node.

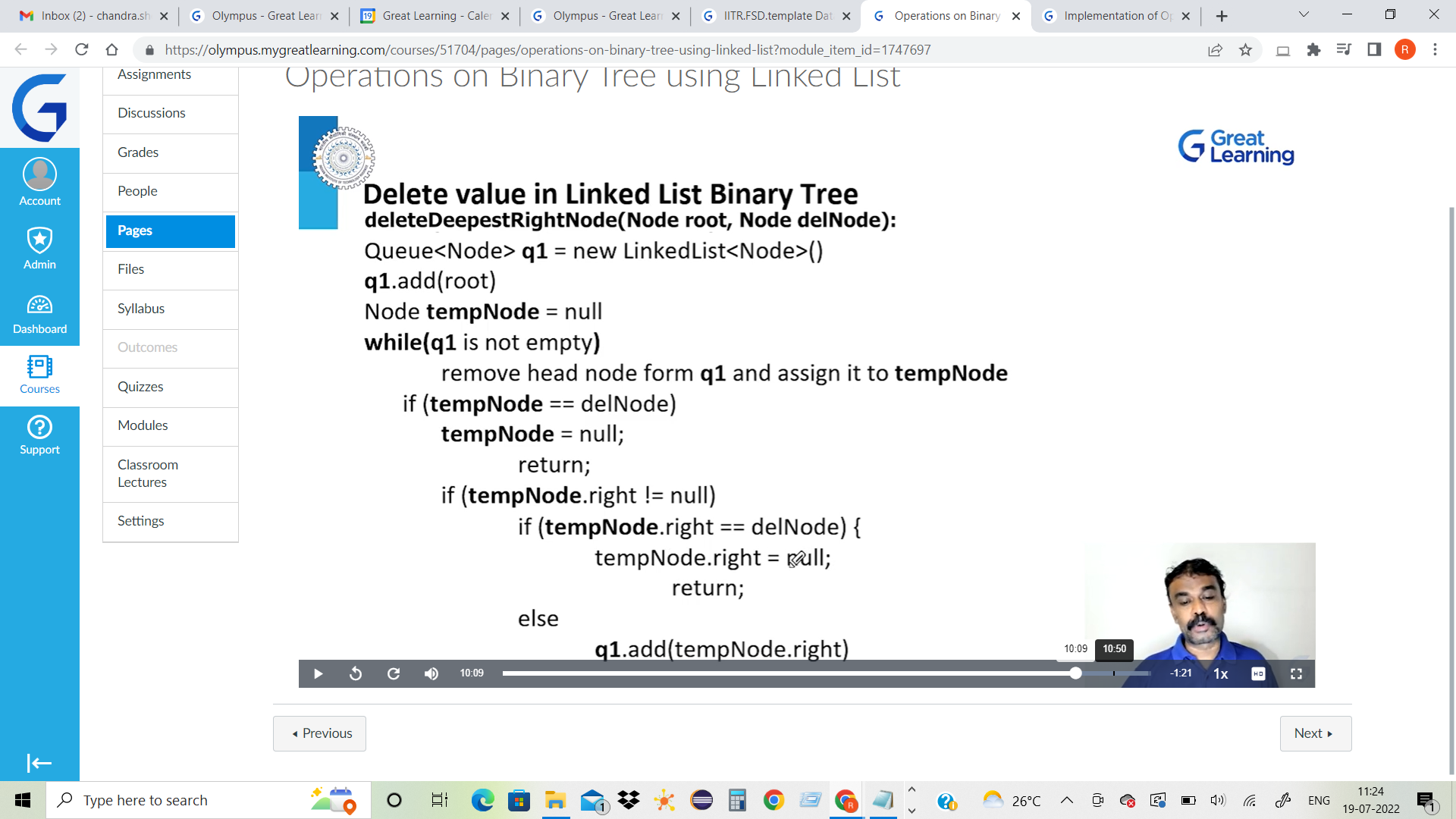


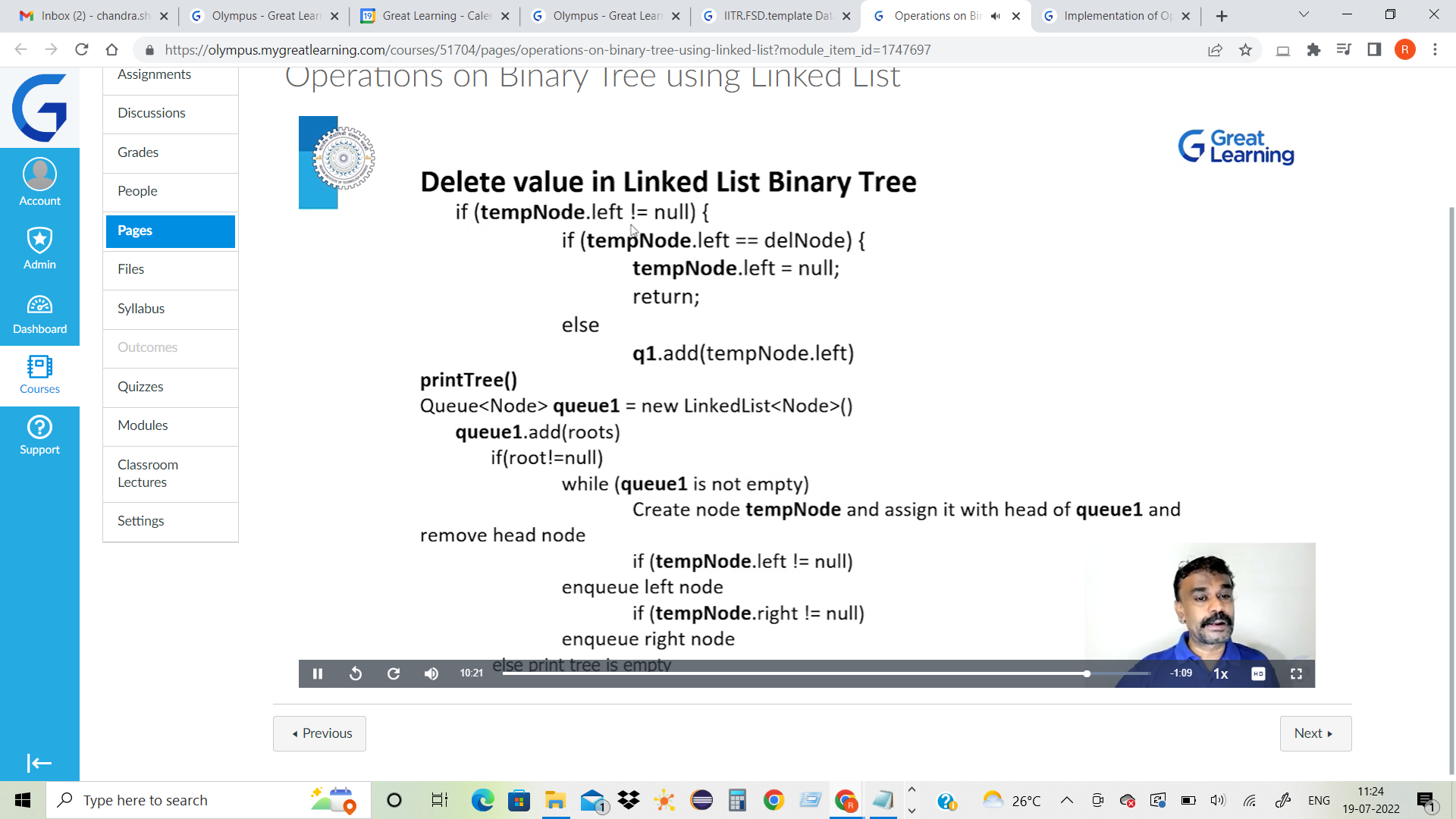
ALGORITHM WOULD BE



That is if the root node is to be deleted and if that is the only node , that would be deleted.







public void deleteNode(int key) {

if (root == null)

return;

if (root.left == null && root.right == null) {

if (root.key == key) {

root = null;

return;

} else

return;

}

Queue<Node> q = new LinkedList<Node>();

q.add(root);

Node temp = null, keyNode = null;

while (!q.isEmpty()) {

temp = q.peek();

q.remove();

if (temp.key == key)

keyNode = temp; // Found the deletion node

if (temp.left != null)

q.add(temp.left);

if (temp.right != null)

q.add(temp.right);

}

if (keyNode != null) {

// temp=findAndDeleteRightmostNode(root);

int x = temp.key; // the last node searched in the above loop is the deepest rightmost node

delete(root, temp); // delete the deepest rightmost node

keyNode.key = x; // replace found node key value with deleted value

}

}

In the above part of the deleteNode() function

The below PART is to delete the root Node when the element that is to **be deleted is Root itself..**

if (root.left == null && root.right == null) {

if (root.key == key) {

root = null;

return;

-------------------------------------

Otherwise it means there are nodes in that case ,then the following part is to be executed

Queue<Node> q = new LinkedList<Node>();

q.add(root);

Node temp = null, keyNode = null;

while (!q.isEmpty()) {

temp = q.peek();

q.remove();

if (temp.key == key)

keyNode = temp; // Found the deletion node

if (temp.left != null)

q.add(temp.left);

if (temp.right != null)

q.add(temp.right);

When nodes are existing, i.e the Queue is not empty ,

temp = made to point to the Head of the queue

Head node is removed and make the temp point to the node that is to be deleted until the node that is to be deleted is found, adding the left and right nodes to the tree maintaining the structure of the tree

Next Part of the code does

int x = temp.key; // the last node searched in the above loop is the deepest rightmost node

delete(root, temp); // delete the deepest rightmost node

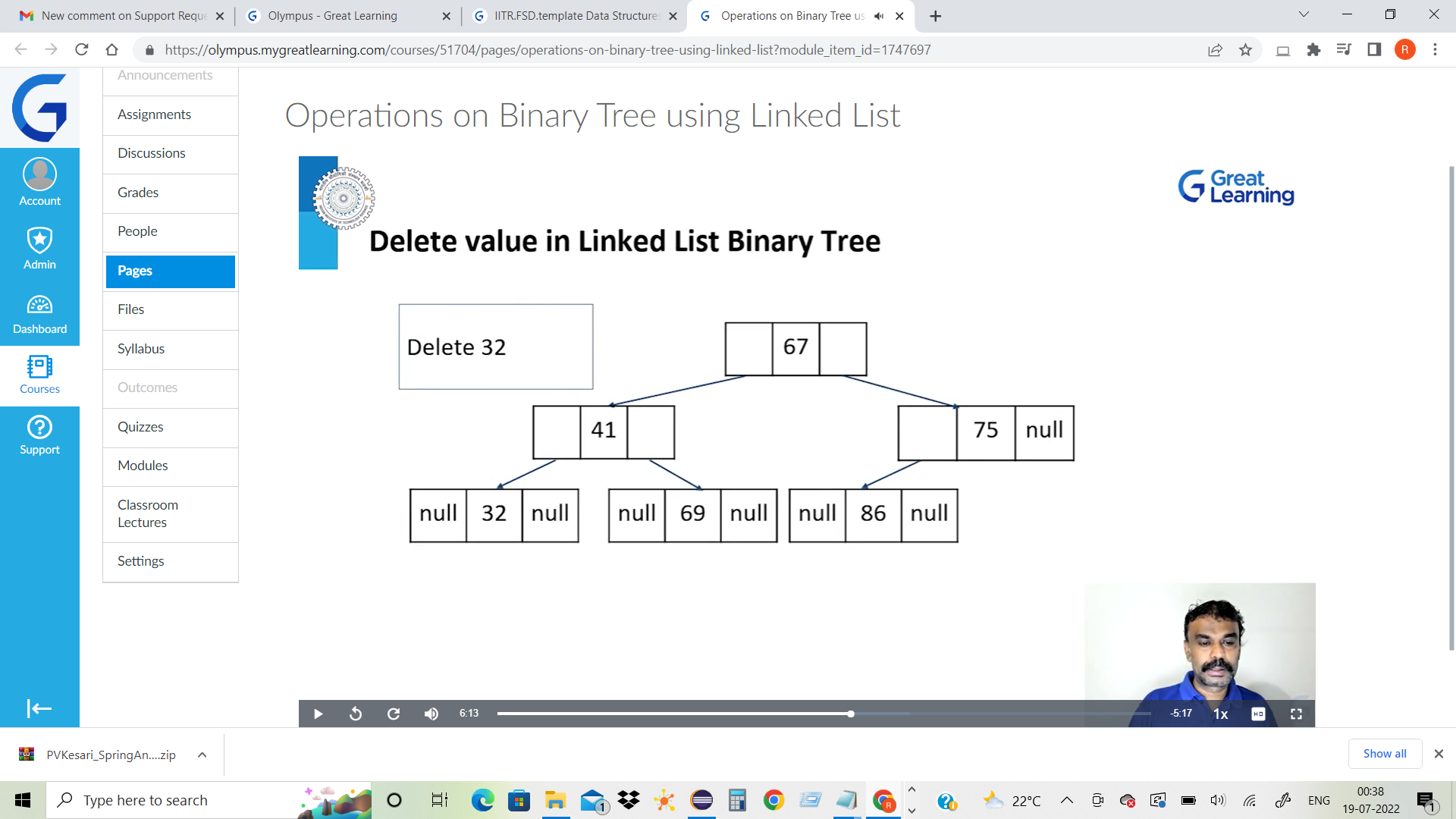
keyNode.key = x; // replace found node key value with deleted value

The deepest rightmost node is found and deleted , before which the data part of the deppest right most node is transferred to x , using keyNode = x earlier we made the keynode is pointing the node that is to be deleted.

Thus the rule

Find the deepest rightmost node, replace it with the deletion node and delete the deepest rightmost node

Is also complied upon.



Next if we wish to delete 41 ; Now the Right most node is 86, replace 41 with 86 and delete the rightmost node in the deepest