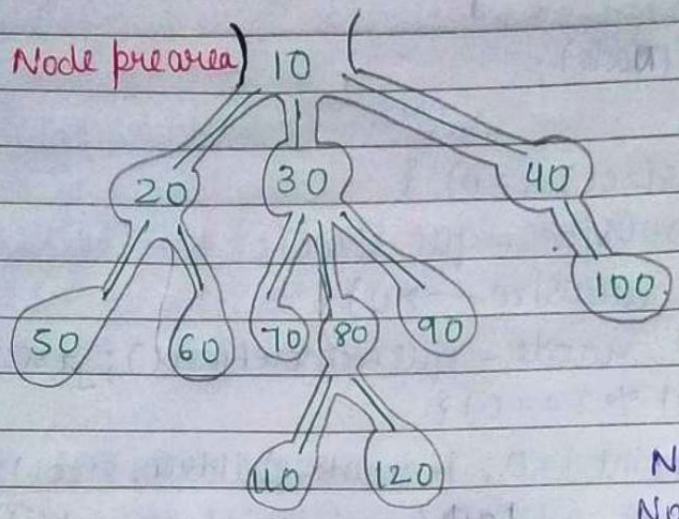
6-Jan-22

Date
DELTA Pg No

1. Generic Tree - Traversals (pre-order, post-order)



Node pre
Node post
Edge pre
Edge post

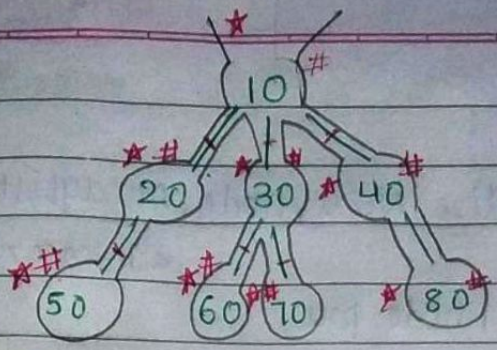
```
// node pre

// loop, before call is edge pre, after call is edge post.

// node post.
```

```
public static void traversals (Node node) {
//node pre
    soutln("Node pre" + node.data);

//loop, before call is edge pre, after call is edge post.
    for(Node child : node.children) {
        soutln("Edge Pre"+ node.data +"--"+ child.data);
        traversals(child);
        soutln("Edge Post" + node.data + "--" + child.data);
    }
    soutlnp("Node Post" + node.data);
//node post
}
```



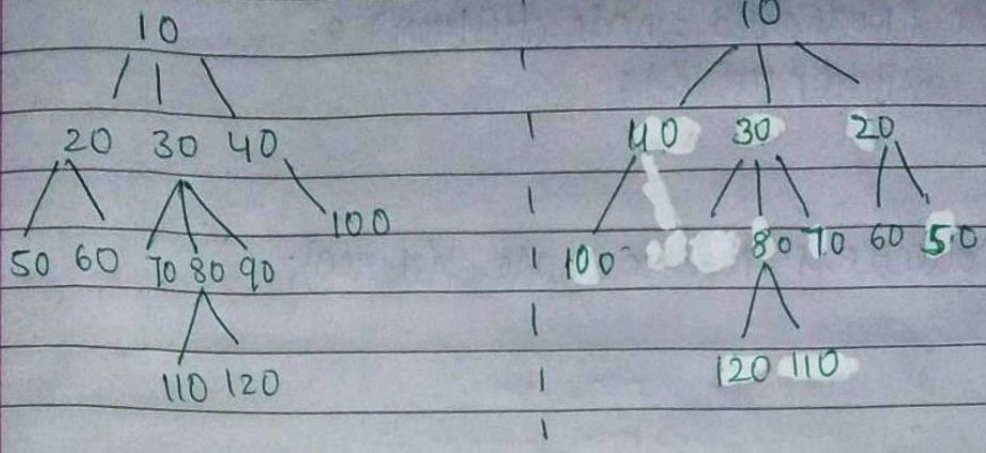
Node pre 10
Edge pre 10-20
Node pre 20
Edge pre 20-50
Node pre 50
Node post 50
Edge post 20-50
Node post 20
Edge post 10-20
Edge pre 10-30
Node pre 30
Edge pre 30-60
Node pre 60
Node post 60
Edge post 30-60
Edge pre 30-70
Node pre 70
Node post 70
Edge post 30-70
Node post 30
Edge post 10-30
Edge pre 10-40
Node pre 40
Edge pre 40-80
Node pre 80
Node post 80
Edge post 40-80
Node post 40
Edge post 10-40
Node post 10.

2. Serialize.

```
public static void serialize(Node node, ArrayList<Integer> list){
list.add(node.data); // node pre
for(Node child : node.children){
    serialize(child, list);
}
list.add(-1);   // node post.
}
public static void main(String[]args){
int[]arr={10,20,50,-1,60,-1,-1,30,70,-1,80,
          110,-1,120,-1,-1,90,-1,-1,40,100,-1,-1,-1};
Node root = construct(arr);
ArrayList<Integer> list = new ArrayList<>();
serialize(root, list);
south(list);
}
```

Date
DELTA Pg No.

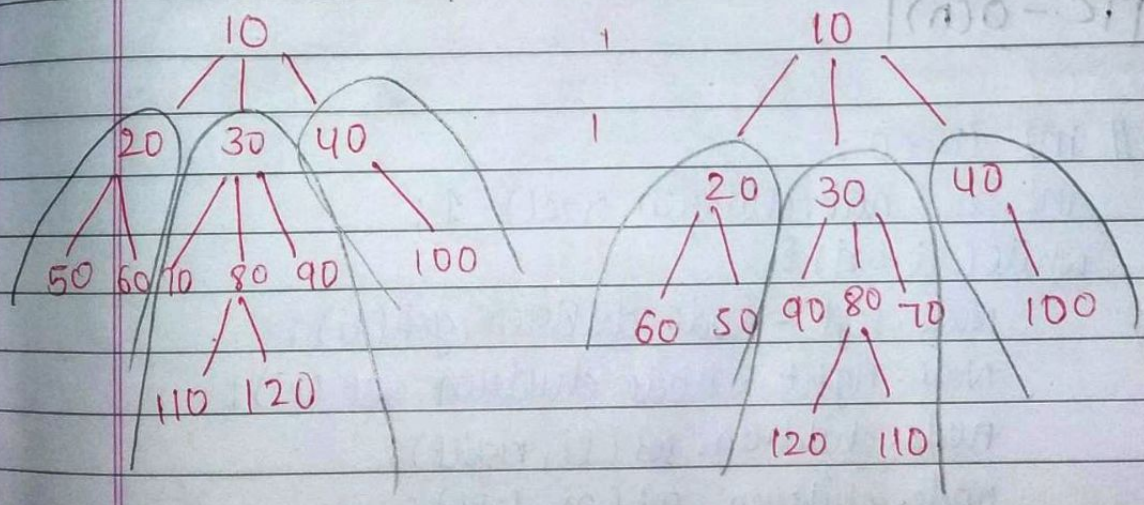
3. Mirror



faith - jisko mirror hone ko keh diya vo mirror hojaega

```
public static void mirror (Node node) {

    for (Node child : node.children) {
        mirror (child);
    }
}
```

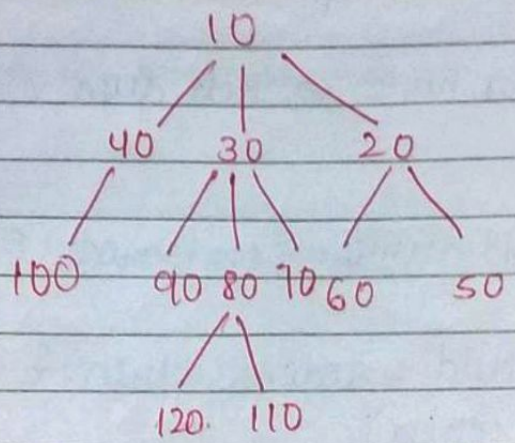


```
public static void mirror(Node node) {

    for (Node child : node.children) {
        mirror(child);
    }

    Collections.reverse(node.children);
}
```

Traversal

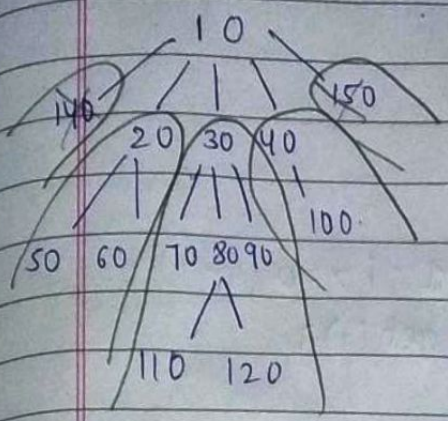


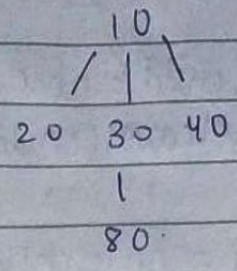
Collections.reverse - ArrayList ko reverse karta h
↳ is a class

TC - O(n)

```
// int li = 0;
   int ri = node.children.size() - 1;
   while (li < ri) {
       Node.left = node.children.get(li);
       Node.right = node.children.get(ri);
       node.children.set(li, right);
       node.children.set(ri, left);
       li++;
       ri--;
   }
```

4. Remove Leaves in Generic Tree -



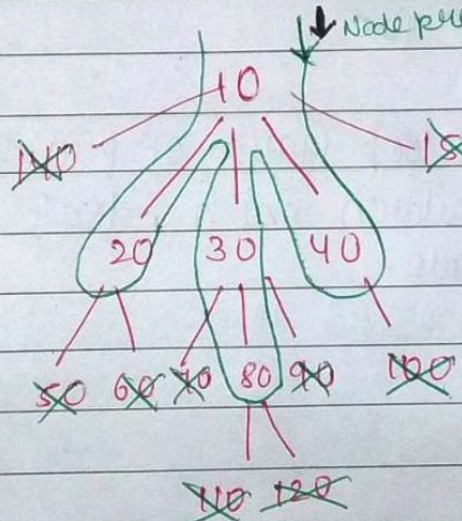


→ • remove your leaves 1st.
→ • request the children.

```
public static void removeLeaves (Node node) {
// remove your own leaves
for(int i= node.children.size()-1; i>=0; i--) {
    Node child = node.children.get(i);

    if(child.children.size()==0) {
     node.children.remove(i);
     }
}
```

Node pre



(150 ke children nhi h to remove)

```
// request the children.
for (Node child : node.children) {
    removeLeaves(child);
}
}
```

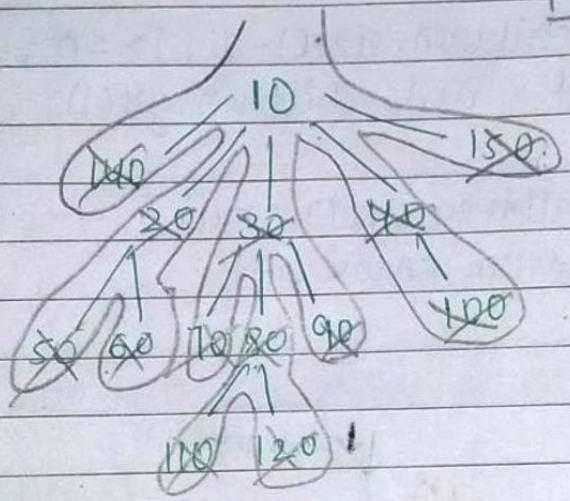
→ Agar removeleaves upar hota to. (post order)

```
for(Node child : node.children) {
    removeLeaves(child);
}
for( int i = node.children.size()-1 ; i>=0 ; i--) {
    Node child = node.children.get(i);

    if(child.children.size()==0) {
        node.children.remove(i);
    }
}
```

3.

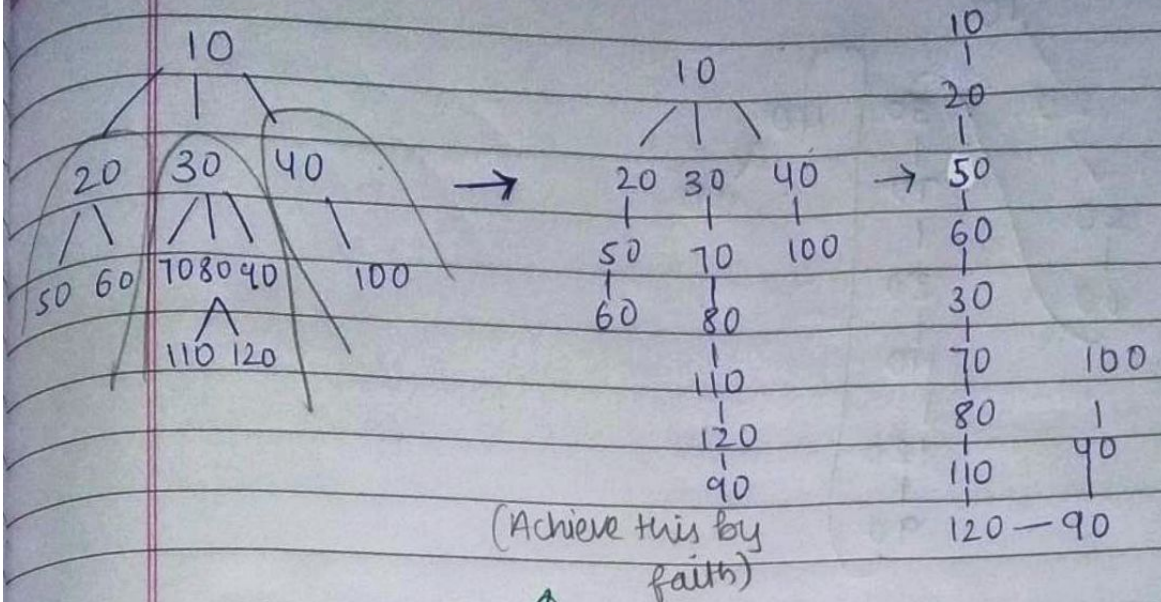
⇒ 10



→ Saare child me loop lgaya or fir dekhna child or child me children nhi h, leaf type ka h to remove krdo.

(i represent children's address)

5. Linearize A Generic Tree -

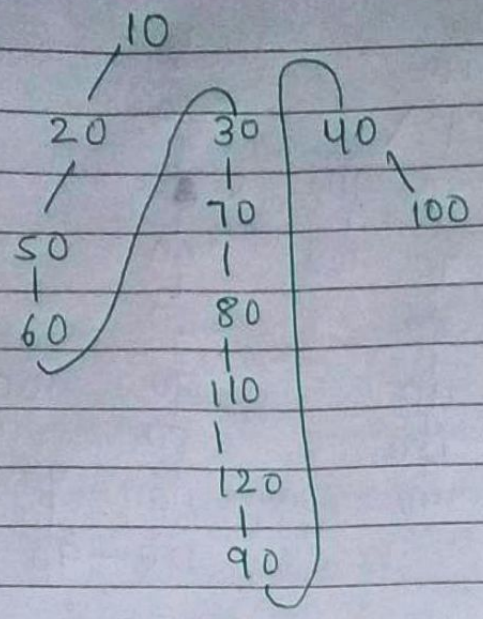



```
for(Node child : node.children){
    linearize(child);
}
```

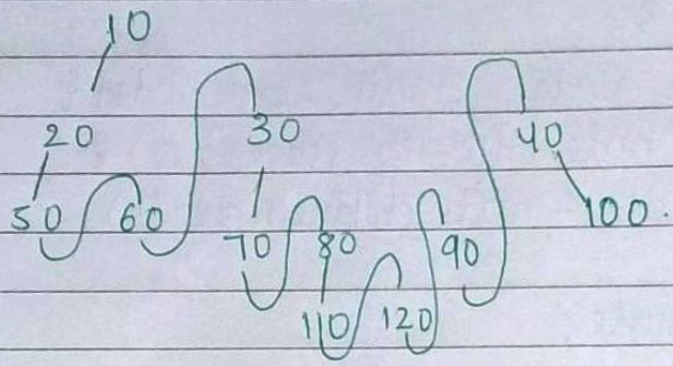
```
public static Node getTail(Node node){
    while(node.children.size()>0){
        node = node.children.get(0);
    }
    return node;
}
public static void linearize(Node node){
    for(Node child : node.children){
        linearize(child);
    }
    //post order
    while(node.children.size()>1){
        Node last = node.children.remove(node.children.size()-1);
        Node slast = node.children.get(node.children.size()-1);
        Node slastkiTail = getTail(slast);
        slastkiTail.children.add(last);
    }
}
```

$O(n^2)$

high level



low level



HW Linearize in O(n)

Hint - w/o void., w/o gettail.