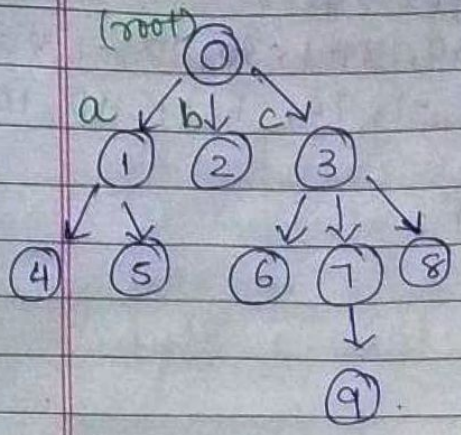
5-JAN-2022

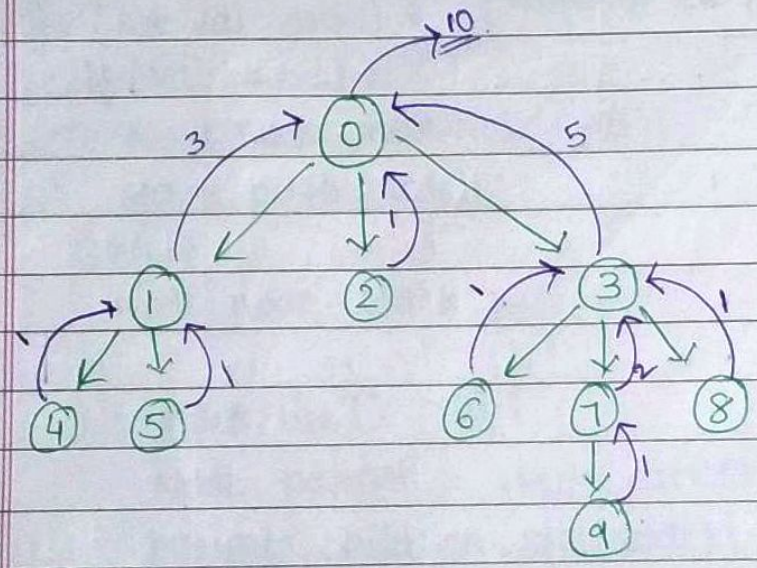
Generic tree -



→ a + b + c + 1

1. Size of Generic Tree.

```
public static int size(Node node) {
    int size = 0;
    for(Node child : node.children) {
        size += size(child);
    }
    return size + 1;
}
```



5, —	0+1
4, —	0+1
1, XX	0+1+1
0, X X 1	0+3

root

child

← Empty ArrayList · Qki 4 ka koi child nhi h)

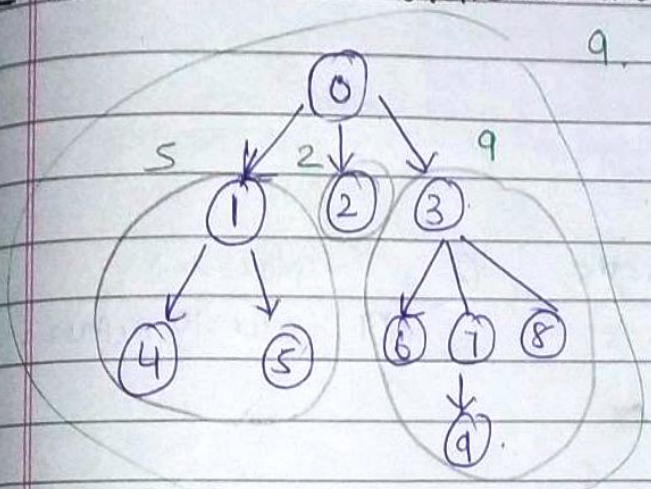
← Size.

8,	0+1
9,	0+1
7, X	0+1
6	0+1
3, XXX	0+1+2+1+1
0, XX 1	0+3+1+5+1

→ 10.

2,	0+1
0, X X 1	0+3+1

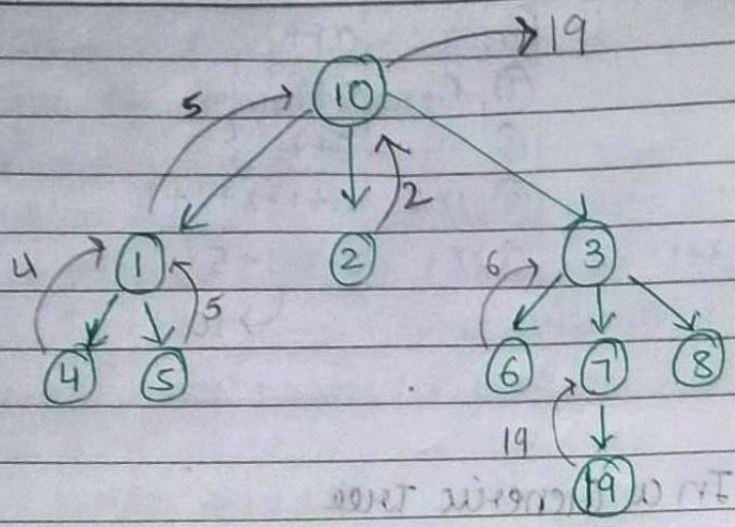
2. Maximum In a Generic Tree.



```
public static int max (Node node) {
    int maxChild = node.data;
    for(Node child : node.children) {
        int recAns = max(child);
        maxchild = Math.max ( recAns, maxChild);
}.
return maxChild;
}.
```

5 , 10

19 , 7



(5)	max = 5		
(4)	~~max = 4~~		
(1), ✓✓	max = ~~15~~, recAns = ~~4~~ 5	(2)	max = 2 ,
(10), ✓✓✓	10 , recAns = 5	(10) ✗✗✓	max = 10, recAns = 2

max

(8)	
(19)	
(7) ✓	max = 7 recAns. = 19
(6)	~~max = 6~~ 19
(3) ✗✗✗	max = ~~26~~, recAns = ~~8 19~~ 8
(10), ✗✗✓	max = 10, recAns = 19

10	20	30	40

for (int i : arr) { }

↳ ek ek krke saare var. ko access krlega

FOR EACH LOOP

Date

DELTA Pg No.

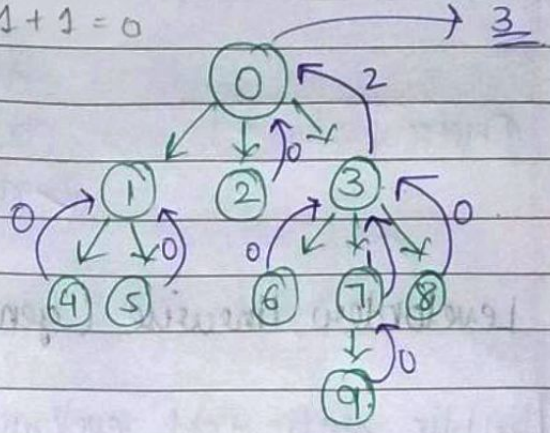
3. Height of a Generic Tree -

```
public static int height(Node node) {
    int height = -1; (assume height - 1)
    for(Node child : node.children) {
        int recAns = height(child); (child ki height mangvai or
        height = Math.max(recAns, height); (uske base pe apni H ko update krna)
    }
    return height + 1;
}
```

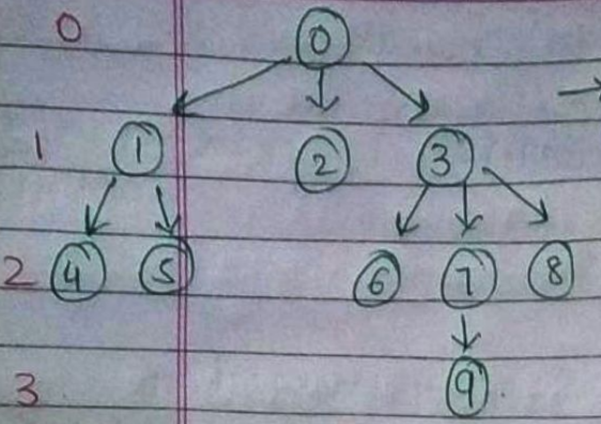
→ Agr 1 Node hogi to H = 0 ; −1 + 1 = 0

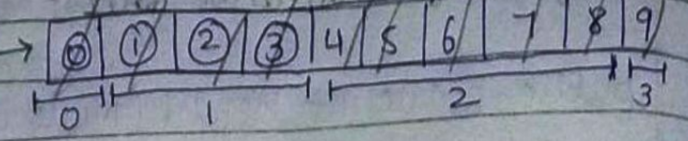
5	~~0~~
4,	~~−1~~ 0
1, X1	~~−1~~, 0
0, X11	~~−1~~ 1

Height



(No need to draw stack)





→ 0
→ 1 2 3
→ 4 5 6 7 8
→ 9.

Queue size = ~~1~~ 0 (remove kra fir print)
(L1) size = ~~3~~ ~~2~~ ~~1~~ 0
(L2) size = ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0
(L3) size = ~~1~~ 0.

Queue : | 1 | 2 | 3 | 4 |

addFirst
addLast
✓ removeFirst

4. Levelorder Linewise (generic Tree).

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

    LinkedList<Node> que = new LinkedList<>();
    que.addLast(node);

    while (que.size() != 0) {
        int currSize = que.size();
                              →(0 se bda ho to1 km krenge)
        while (currSize --> 0) {   >0
            Node rnode = que.removeFirst();
            soutn (rnode.data + " ");
```

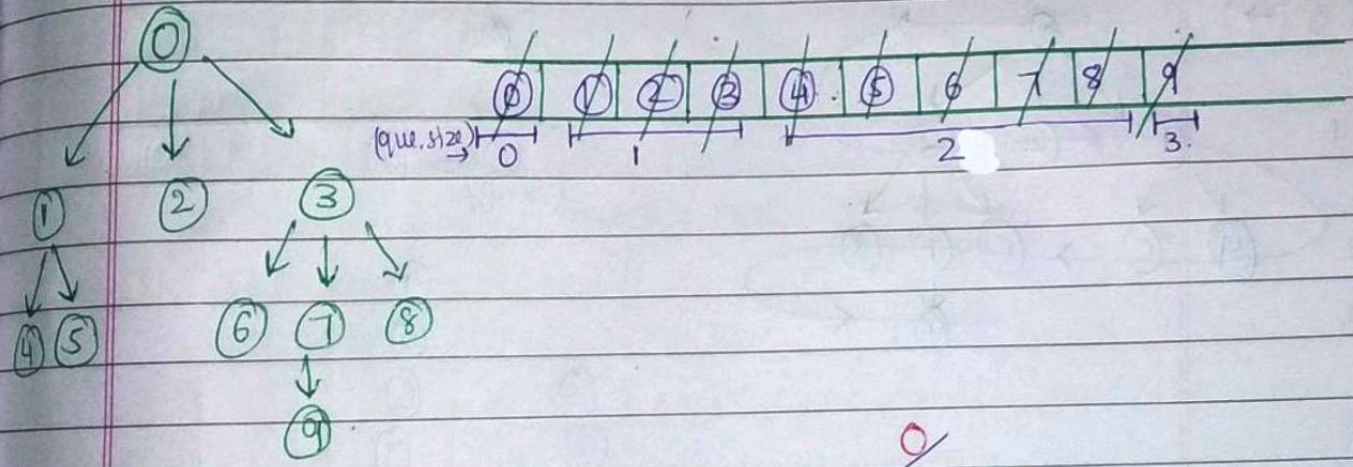
Date
DELTA Pg No.

⇒ Post decrement = [--->]
↳ Phle checking, fir decrement.

addFirst + removeFirst = STACK.
removeFirst + AddLast = QUEUE

```
for (Node child : rnode.children) {
    que.addLast(child);
}
}
solution();
```

TC - O(n) [Oki hr ek node ek baar hi visit hua h] (n-sec)



currSize = ✗0
= ✗✗0
= ✗✗✗0
= ✗0.

0
→ 1 2 3
→ 4 5 6 7 8
→ 9

Why NOT ArrayDeque?

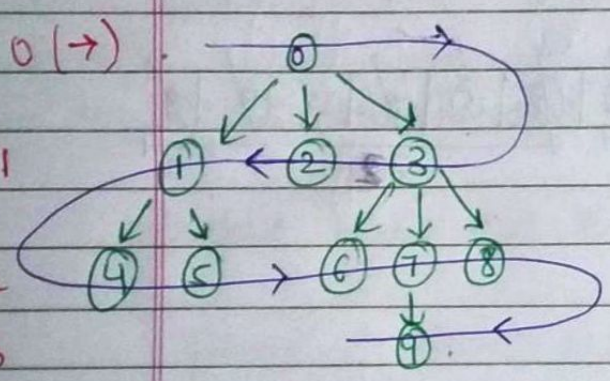
→ Cannot add Null in ArrayDeque, it will give Null pointer exception.

→ AddFirst + RemoveFirst = STACK.
(& Stack)

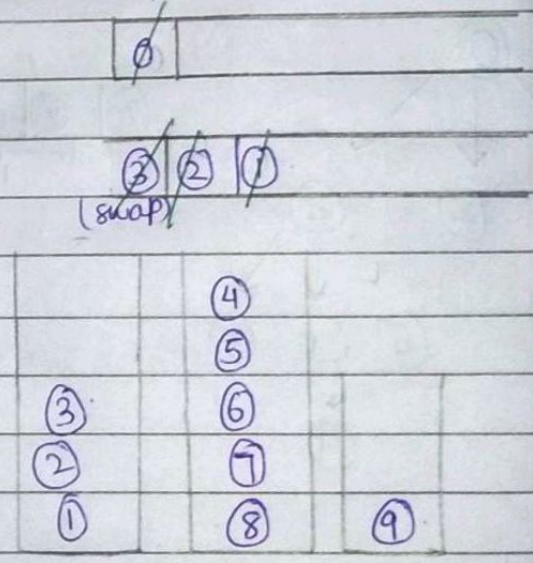
→ LinkedList can be used as linked list, STACK, QUEUE.

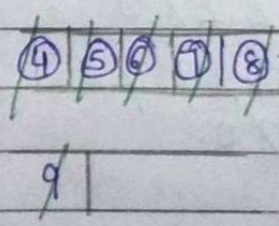
level Even :→ left to right (→)] Child
" Odd :→ right to left (←)] "

5. Levelorder Linewise ZigZag -



0
3 2 1
4 5 6 7 8
9





```
public static void levelOrderLinewiseZZ(Node node) {
    LinkedList<Node> que = new LinkedList<>();
    LinkedList<Node> st  = new LinkedList<>();

    que.addLast(node);
    int level = 0;
    while (que.size() != 0) {
        int currSize = que.size();
        while (currSize--> 0) {
            Node rnode = que.removeFirst(); System.out.print(rnode.data
                                                + " ");
            if (level % 2 == 0) {
                for(int i=0; i< rnode.children.size(); i++)
                    st.addFirst (rnode.children.get(i));
            } else {
                for (int i = rnode.children.size()-1; i >=0 ; i--)
                    st.addFirst (rnode.children.get (i));
            }
        }
        level ++;
        System.out.println();
        LinkedList<Node> temp = que;
        que = st;
        st = temp;
    }
}
```