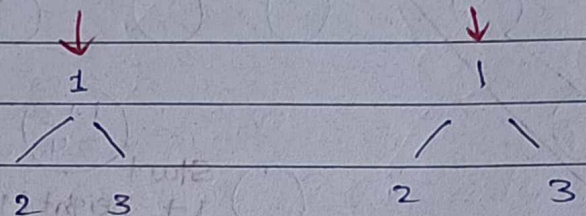
Page No.
Date : / /

Lecture - 41

Trees - Easy problem

problem - 1 : Determine if Two trees are identical

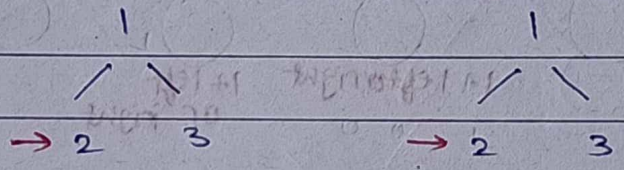
Example 1 :



Start from root

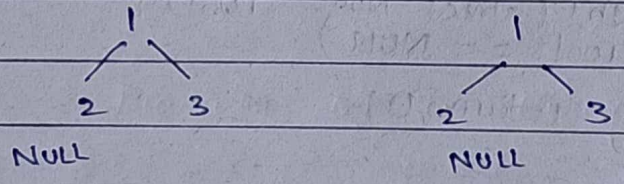
1 = 1 (Equal)

Move to left



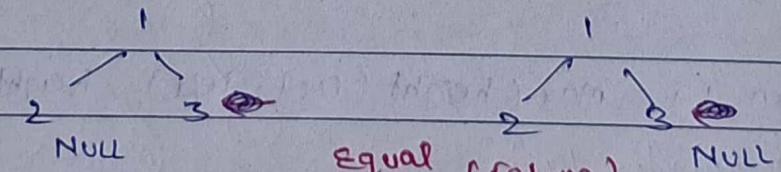
2 = 2 (Equal)

Move to left

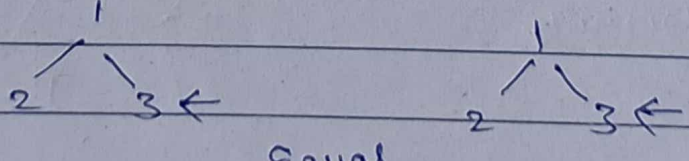


Equal.

Move to right



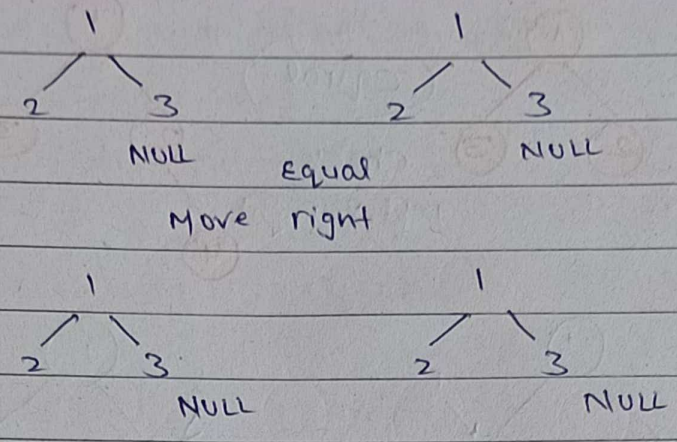
Move right



Equal

Move left.

Teacher's Signature.....



Case:

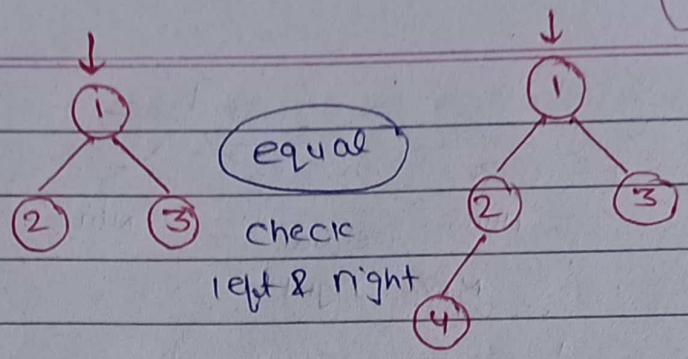
Tree 1	Tree 2		
NULL	Address	→	0
Address	NULL	→	0
NULL	NULL	→	1
Address	Address	→	0/1

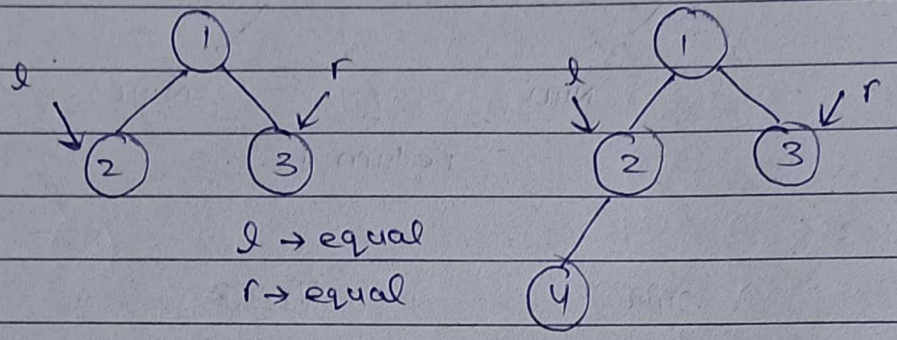
Code:

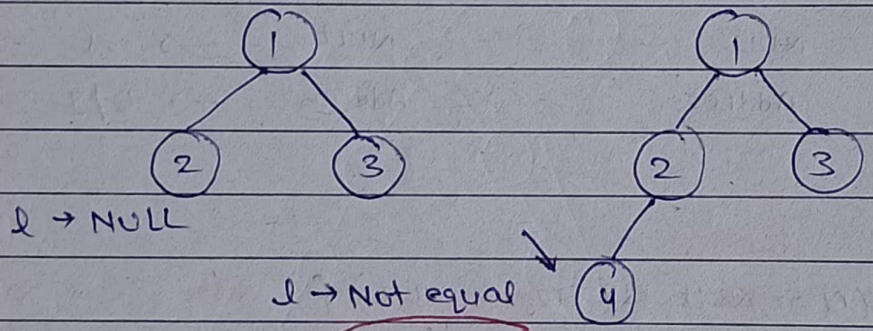
```
if (r1 == NULL && r2 == NULL) {
    return 1;
}
if (r1 && r2 == NULL || r2 && r1 == NULL) {
    return 0;
}
if (r1 -> data != r2 -> data) {
    return 0;
}
return isIdentical(r1->left, r2->left) && isIdentical(r1->right,
                                                      r2->right);
```

Teacher's Signature.....

Example 2

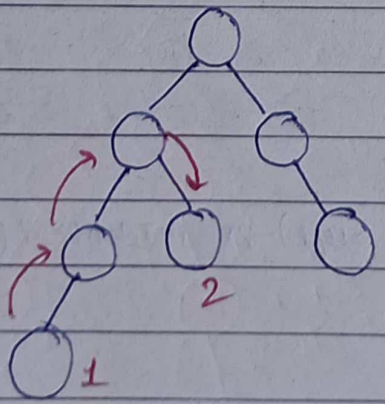






Problem 2 : Diameter of a Binary Tree

How to find diameter :



find diameter b/w 1 & 2 :

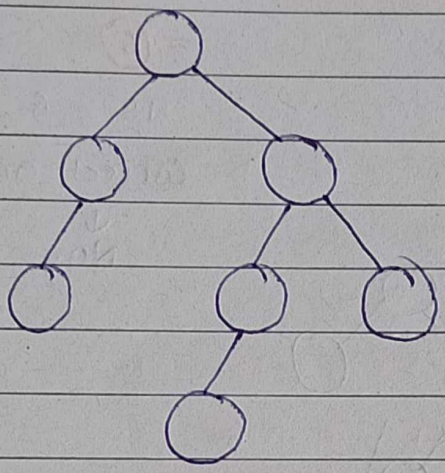
Diameter = 4

Teacher's Signature.....

You task is to find the longest diameter present in Binary tree.

Example:

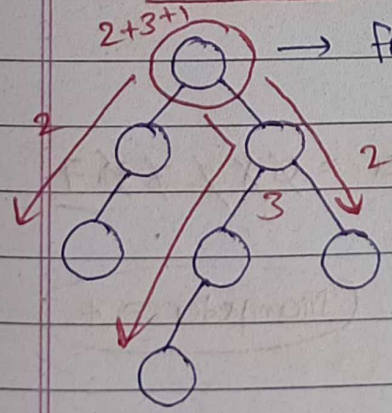
Diameter: 2 Node के बीच Maximum Node include both the two Node.



How to find:
diameter always start & end with leaf Nodes.

find the leaf Node:

Approach 1:



→ from this point we find the heigh from left and right.

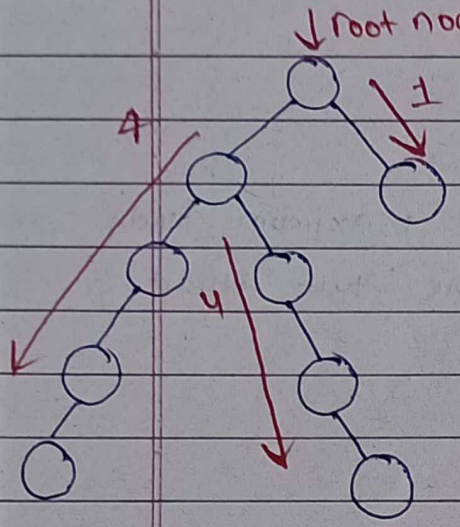
left height = 2
right height = 3

diameter = 1 + 2 + 3
= (6)
↳ Ans.

But can we alway find the correct height from only root.

Teacher's Signature.....

Example 2 :



left height = 4

right height = 1

diameter = 1 + 4 + 1

= (6)

is 6 is the correct answer.

↓

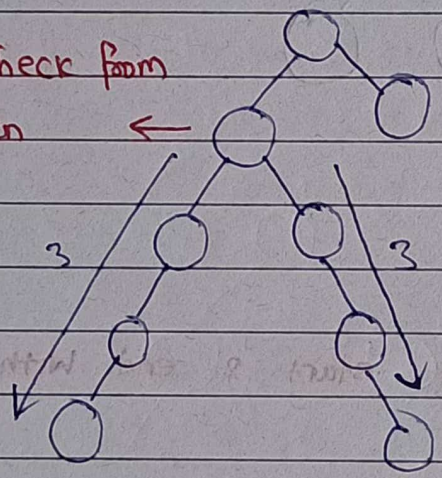
No.

If we check from this point then diameter is

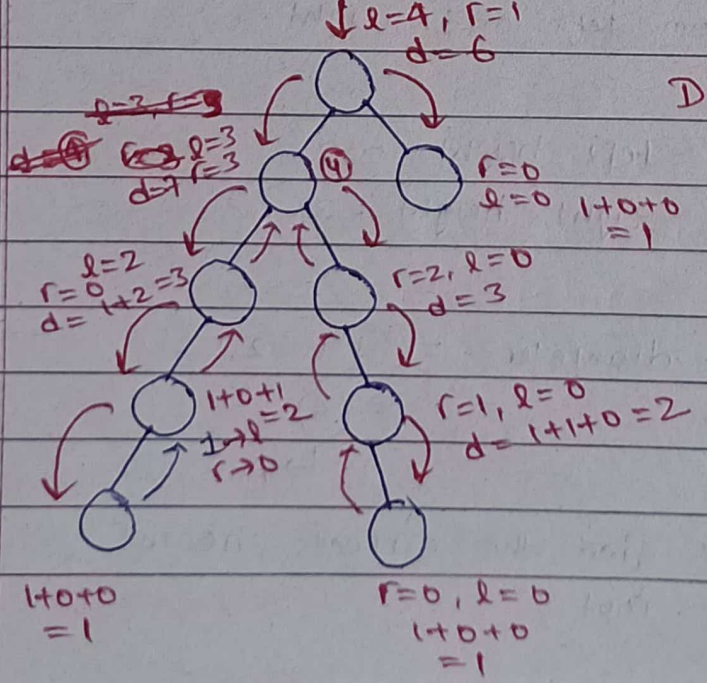
3 + 3 + 1

= (7)

7 > 6



So, we have to check every node for measuring the diameter.



Diameter = ~~0 X 2 3 4 5 6~~ 7

Diameter = 7

Teacher's Signature.....

1. left height
2. right height
3. find max from both and +1 = temp
4. diameter = max (diameter, temp)

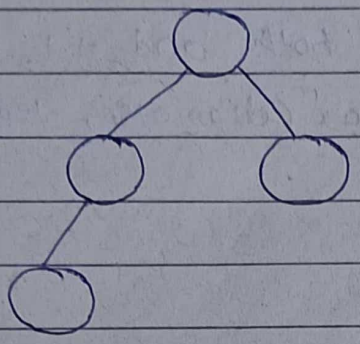
base condition :

either Node Exist or NULL.

Code :

```
int diameter (Node * root) {
        int ans = 0;
        find (root, ans);
        return ans;
}

int find ( Node * root, int & ans) {
        if (! root)
            return 0;

        int left = find (root → left, ans);
        int right = find (root → right, ans);

        ans = max ( ans, 1 + left + right );

        return 1 + max (left, right);

}
```

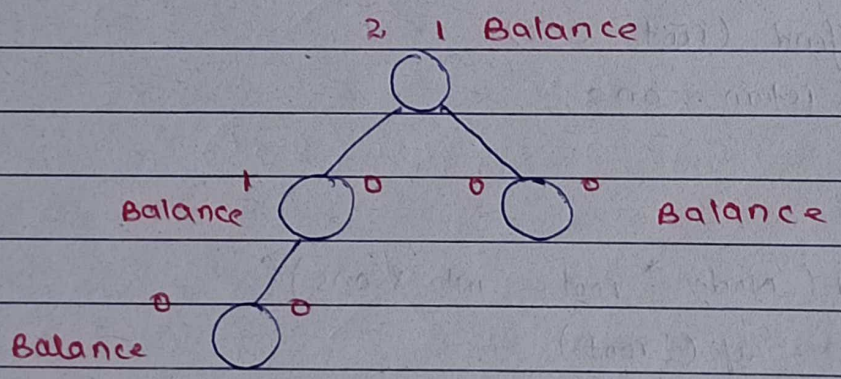
Problem 3 : Check for balanced Tree



difference of left & right subtree not > 1.

Approach :

1. calculate left side height
2. calculate right " "
3. find the diff.
   ↳ if > 1 return 0.



Ek answer variable

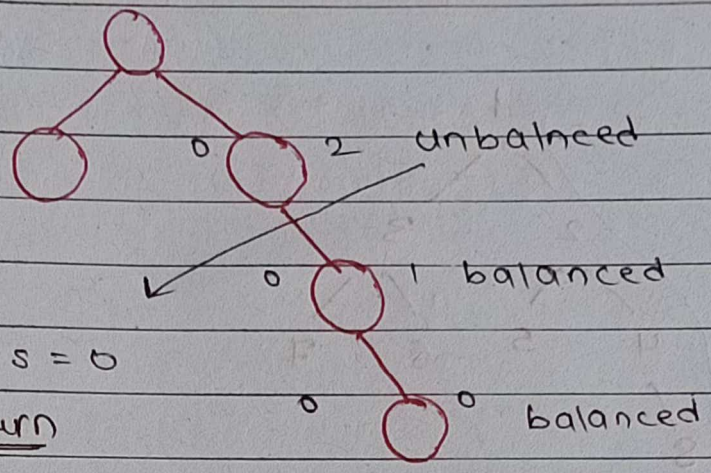
~~ans → 0~~

ans = 1;

अगर एक भी Node balance नहीं रहेगा तो whi pe turant 0 kar denge or return कर देंगे,

Page No.:
Date: / /

Example : 2

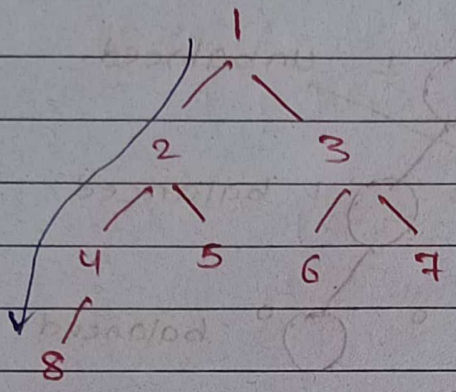


Code :

```
bool isBalanced (Node * root) {
        bool ans = 1;
        findHeight (root, ans);
        return ans;
    }

int findHeight (Node * root, bool &ans) {
        if (root == NULL)
            return 0;

        int left = findHeight (root → left, ans);
        int right = findHeight (root → right, ans);

        if (abs (left - right) > 1)
            ans = 0;

        return 1 + max (left, right);

}
```

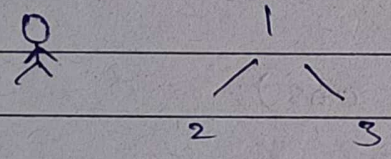
Teacher's Signature.....

Problem 4: Left View of Binary Tree

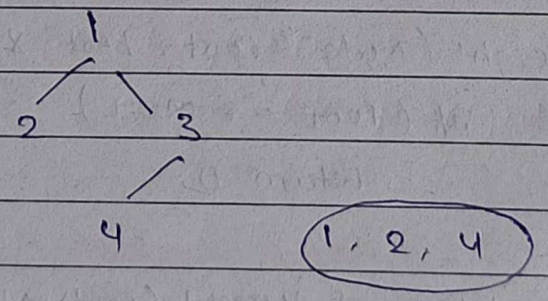


Left View : 1 2 4 8

Visited from left side.



(1, 2) output



How to solve this?

↳ recursion

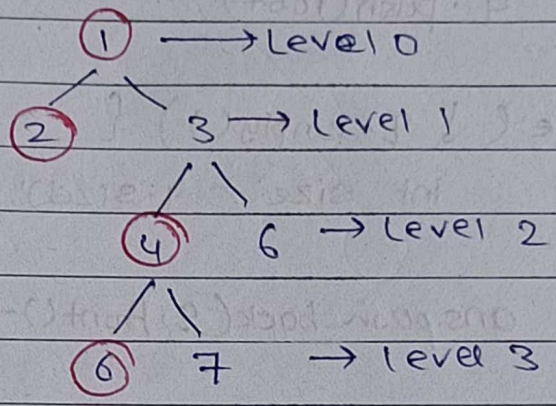
↳ iterative.

Teacher's Signature.....

Page No.:
Date: / /

(1) Iterative approach

Har level ka first Node



1, 2, 4, 6

check all level and return 1st Node of each level from left.

Queue | 2, 3

1 visited
↳ said 2, 3

Ans : 1, 2

only first can be taken in answer. Neglect all other.

first element → 2
3 can't answer

| 4, 5

Ans : 1, 2, 4

5 can't answer

| 6, 7

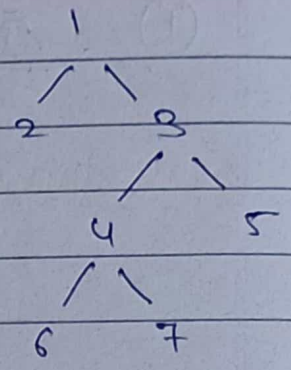
Ans : 1, 2, 4, 6

7 can't answer

Teacher's Signature.....

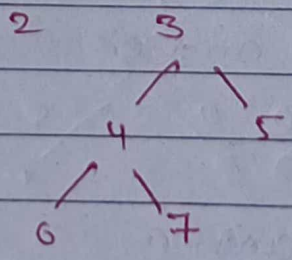
Code:

```
Vector <int> ans;
if (root == NULL) return ans;
queue <Node *> q;
q.push(root);

while (!q.empty()) {
    int size = q.size(); // tells the no. of element present on that level.
    ans.push_back(q.front()->data);

    while (size--) {
        Node *temp = q.front();
        q.pop();

        if (temp->left) {
            q.push(temp->left);
        }
        if (temp->right)
            q.push(temp->right);
    }
}
return ans;
}
```



Explanation (dry run)

(1) q.push(root) → q | 1



ans = { 1 }

q.empty() → NO

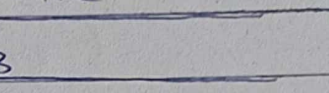
size = 1

ans.push(q.front()) = (1)

Teacher's Signature.....

temp = 1 — 1 deleted.

q.pop();



temp→left = ~~NULL~~ 2

↳ ~~no push~~ push to queue

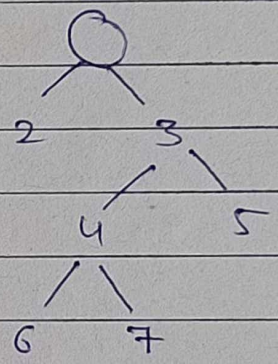
temp→right = 3 push

loop end

again loop start

Size = 2

ans.push(2)



ans = {1, 2 }

temp = 2

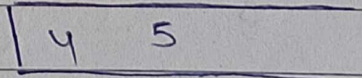
2 deleted from q. Size = 1

temp→left NULL

temp→right NULL

temp = 3

3 deleted



temp→left = 4

temp→right = 5

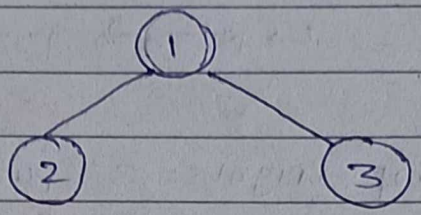
Size = 0

loop end

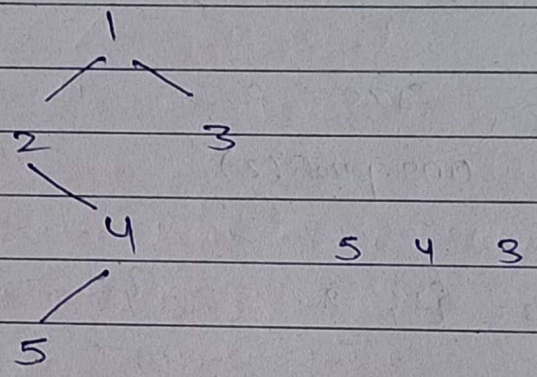
Teacher's Signature.....

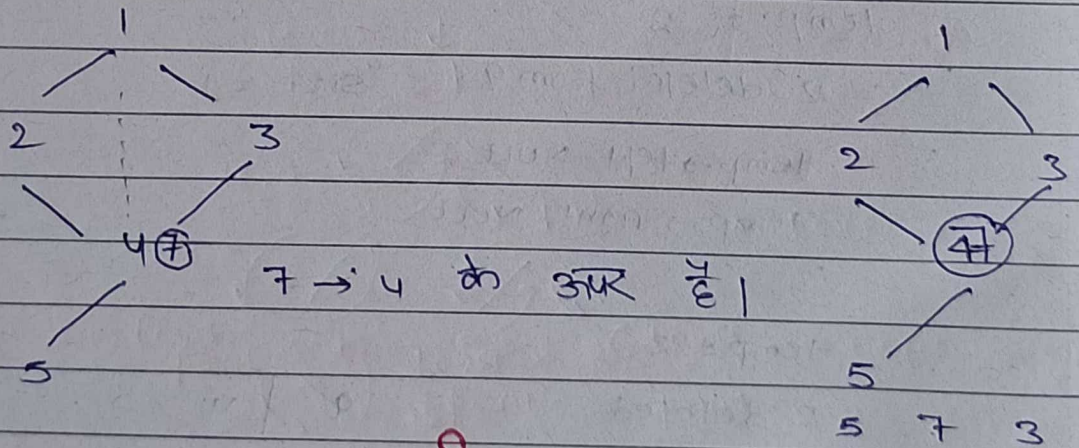
By this process we can Complete the Question.

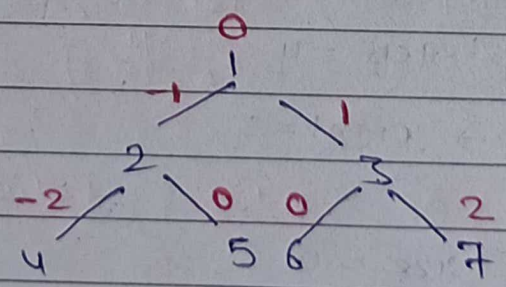
Problem 5: Bottom View of Binary Tree



Ans :- 2 1 3





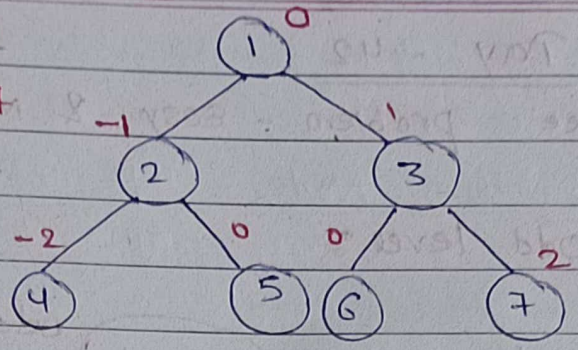


-2 -1 0 1 2

4 2 6 3 7

Teacher's Signature.....

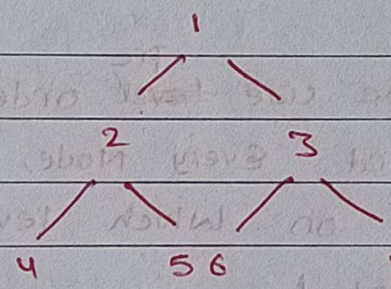
-2	-1	0	1	2
4	2	~~4~~ 5 6	3	7



q : ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~

q : -1 1 -2 0 0 2

Level order.



0
1

q : ~~1~~ 2 3

0

q : ~~2~~ 3 4 5 | 1 0
-1 1 -2 0 | 2 1

q : ~~3~~ 4 5 6 7 | -1 0 1
1 -2 0 0 2 | 2 1 3

q : ~~4~~ ~~5~~ ~~6~~ ~~7~~ | -2 -1 0 1 2
-2 0 0 2 | 4 2 ~~4~~ 5 6 3 7

Ans: 4 2 6 3 7

Teacher's Signature.....