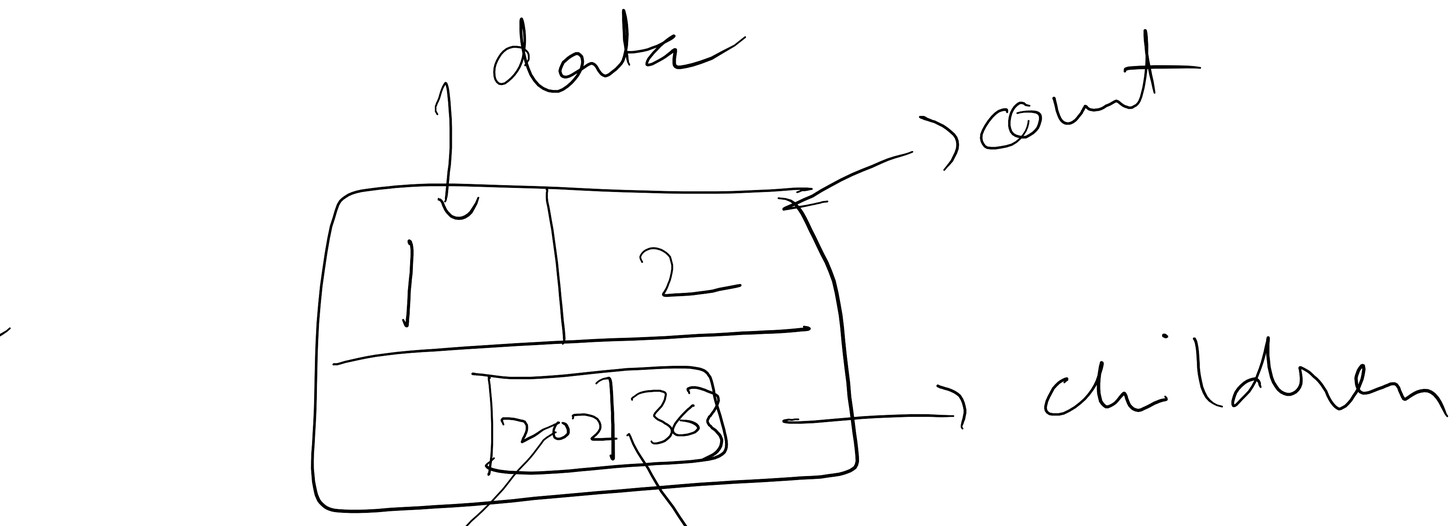
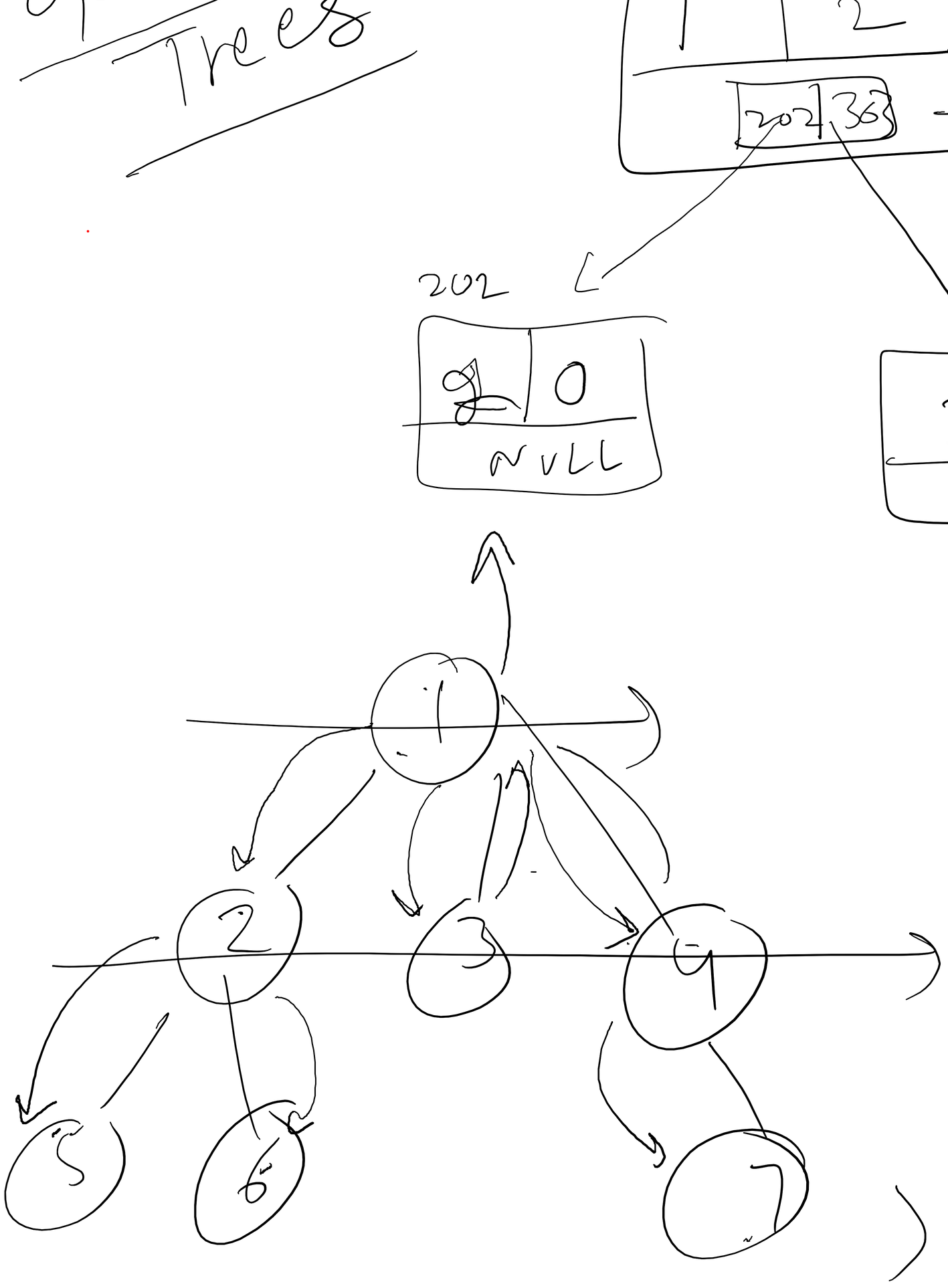
Generic Trees

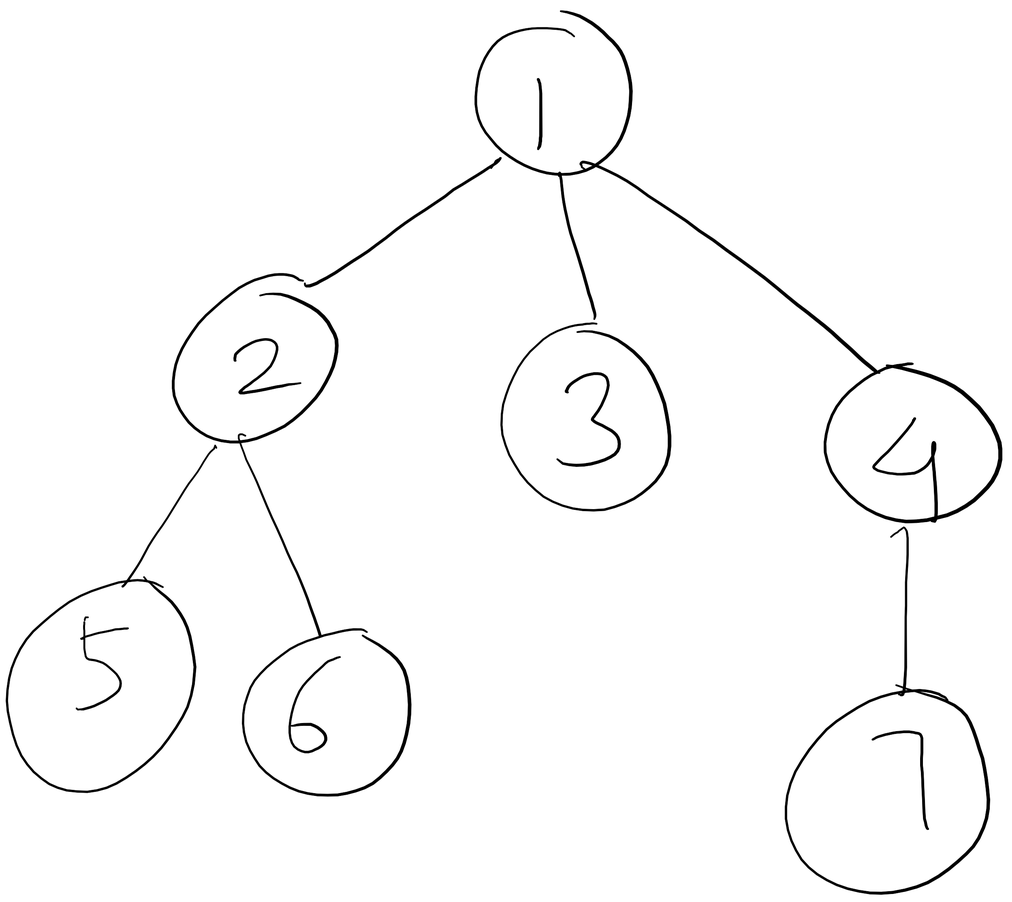




1

2 3 4

5 6 7



1 → 2, 3, 4

2 → 5, 6

3 →

4 → 7

5 →

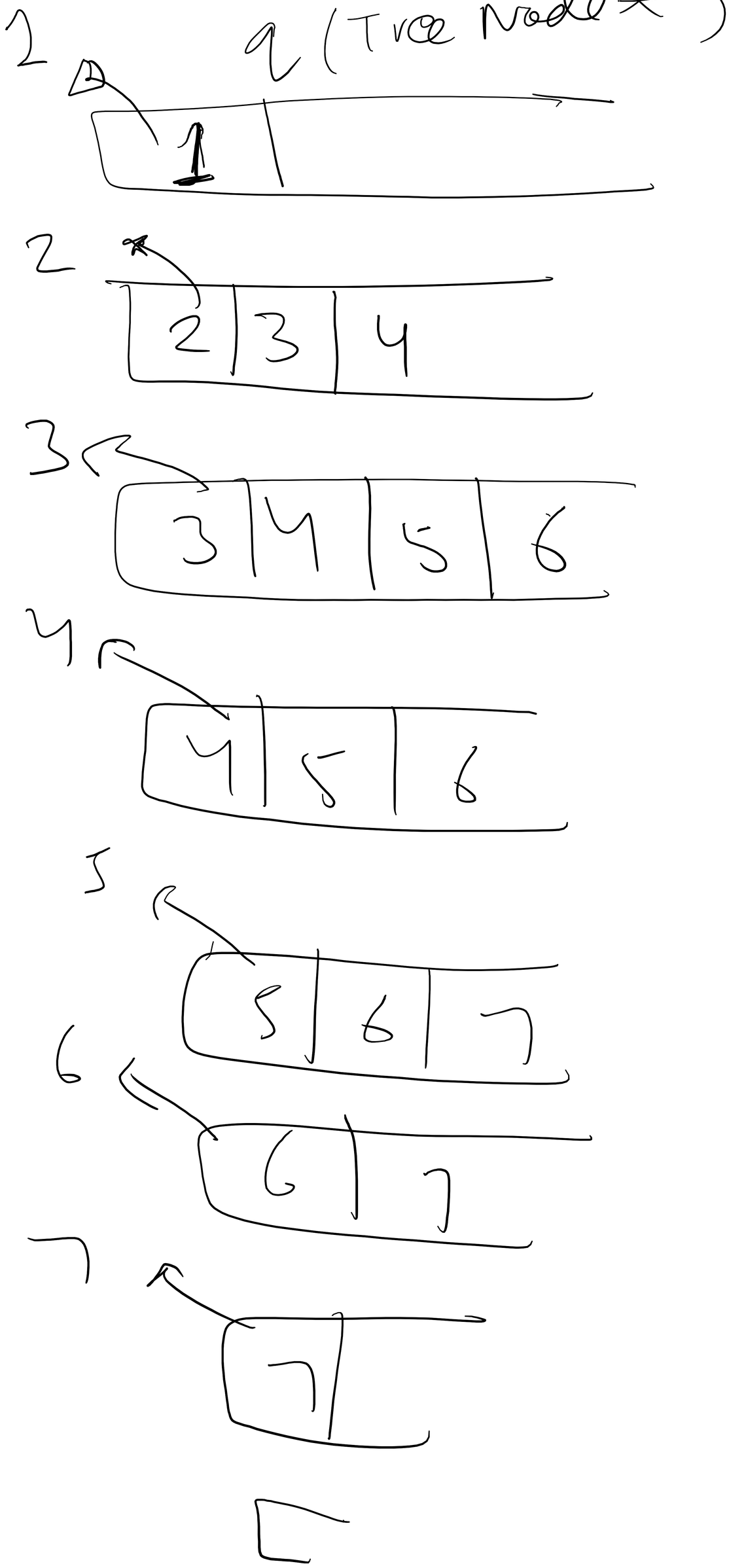
6 →

7 →

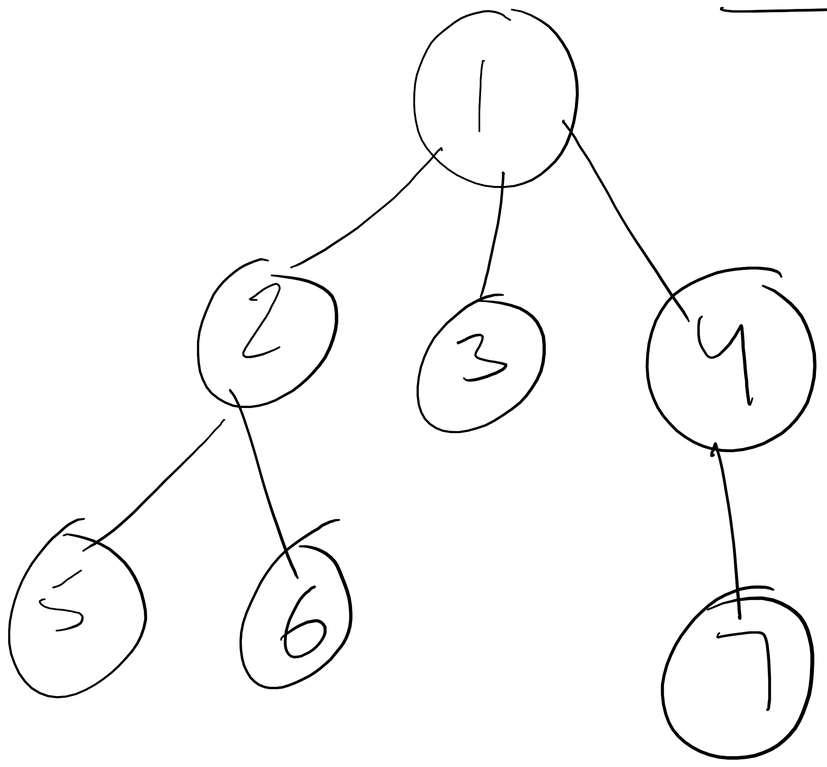
Level Order

Input / Print

q (TreeNode*)



Level Order (Giving lines)



1
2 3 4
5 6

1 | NULL

1

NULL | 2 | 3 | 4

2 | 3 | 4 | NULL

3 | 4 | NULL | 5 | 6

4 | NULL | 5 | 6 | 7

NULL | 5 | 6 | 7 | NULL

5 | 6 | 7 | NULL

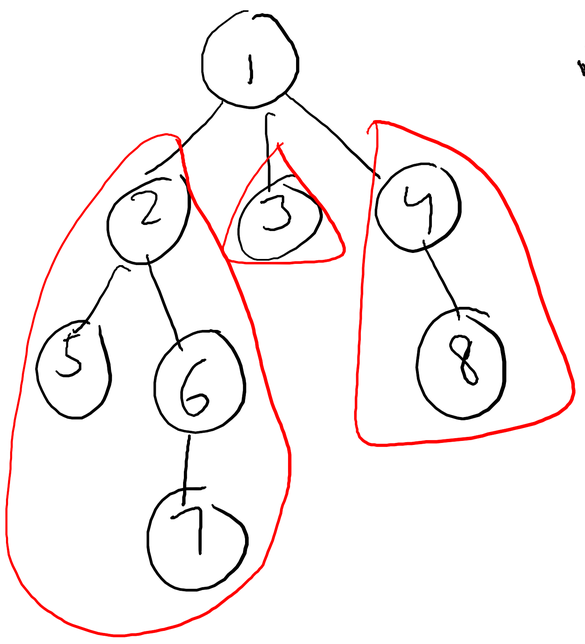
6 | 7 | NULL

7 | NULL

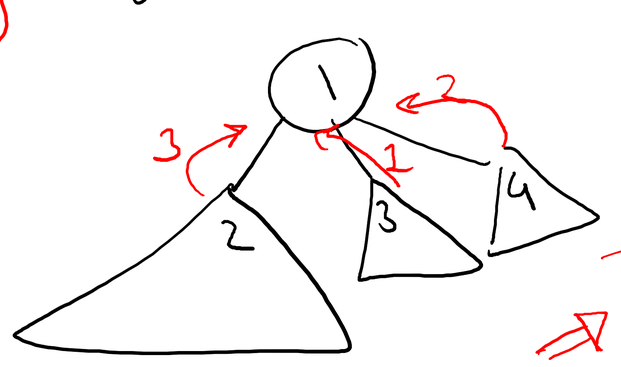
NULL

Don't push another NULL if q is empty

Height of a Tree




```
int Height (TreeNode * root)
{
    // Base Case
    // Rec. Case
}
```



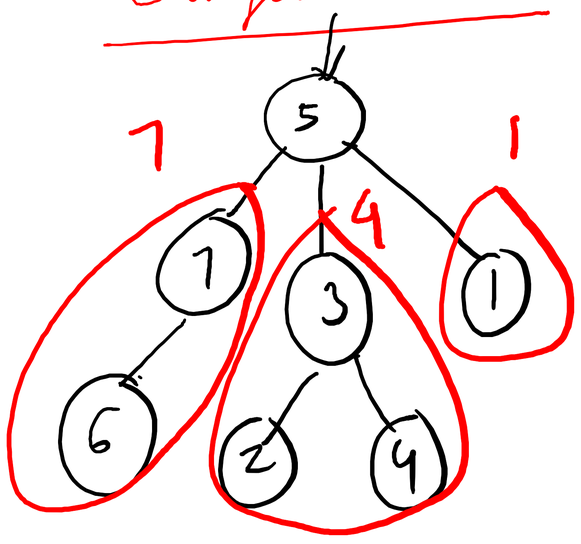
Recursive Case

$H = 1 + \max(3,1,2)$

$= 4$

Max of children/subtrees heights

Largest Node

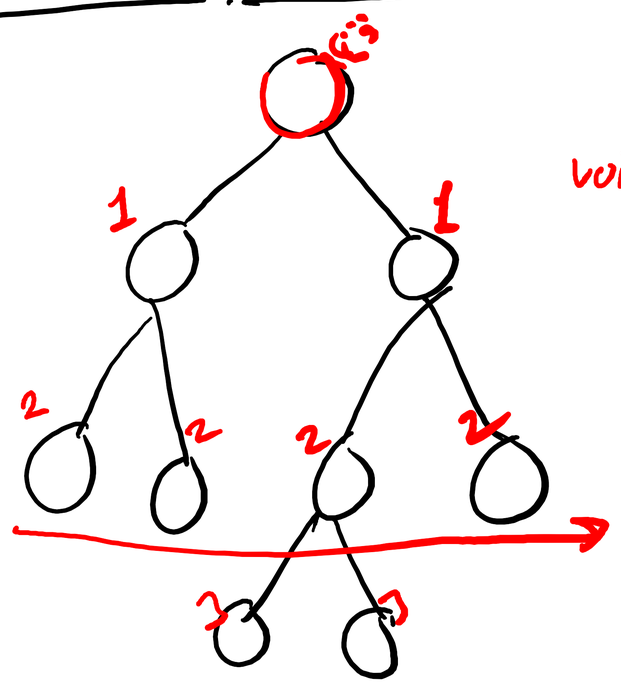


Largest.

→ Find Largest of all subtrees.

→ Compare Root Data and largest found in Subtrees. and Return it.

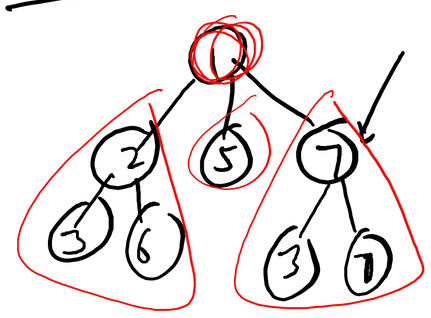
K^{th} Nodes/Level.



Assume k= 2

```
void printk(root, K, c=0)
{
    if(c == k)
    {
        cout<<root->data;
        return;
    }
    if(root == NULL)
        return;
    printk(root->left, k, c+1);
    printk(root->right, k, c+1);
}
```

Node with Max Sum of ith Children & Node



1 = 15

2 = 11

5 = 5

7 = 17

3 = 7

6 = 6

7 = 7

3 = 3