In [1]:

```
# Recursive Python program for level
# order traversal of Binary Tree

class Node:
    def __init__(self, key):
        self.data = key
        self.left = None
        self.right = None


# Function to  print level order traversal of tree
def printLevelOrder(root):
    h = height(root)
    for i in range(1, h+1):
        printCurrentLevel(root, i)


# Print nodes at a current level
def printCurrentLevel(root, level):
    if root is None:
        return
    if level == 1:
        print(root.data, end=" ")
    elif level > 1:
        printCurrentLevel(root.left, level-1)
        printCurrentLevel(root.right, level-1)


# Compute the height of a tree-
def height(node):
    if node is None:
        return 0
    else:
        lheight = height(node.left)
        rheight = height(node.right)
        if lheight > rheight:
            return lheight+1
        else:
            return rheight+1


if __name__ == '__main__':
    root = Node(1)
    root.left = Node(2)
    root.right = Node(3)
    root.left.left = Node(4)
    root.left.right = Node(5)
    printLevelOrder(root)
```

```
1 2 3 4 5
```

In [5]:

```
# Recursive Python program for level
# order traversal of Binary Tree

class Node:
    def __init__(self, key):
        self.data = key
        self.left = None
        self.right = None


# Function to  print level order traversal of tree
def printLevelOrder(root):
    h = height(root)
    for i in range(1, h+1):
        printCurrentLevel(root, i)


# Print nodes at a current level
def printCurrentLevel(root, level):
    if root is None:
        return
    if level == 1:
        print(root.data, end=" ")
    elif level > 1:
        printCurrentLevel(root.left, level-1)
        printCurrentLevel(root.right, level-1)


# Compute the height of a tree-
def height(node):
    if node is None:
        return 0
    else:
        lheight = height(node.left)
        rheight = height(node.right)
        if lheight > rheight:
            return lheight+1
        else:
            return rheight+1


if __name__ == '__main__':
    root = Node(input("Enter root node "))
    root.left = Node(input("Enter left to root node "))
    root.right = Node(input("Enter Right to root node "))
    root.right.left = Node(input("Enter left to Right root node "))
    root.left.left = Node(input("Enter left to left root node "))
    root.left.right = Node(input("Enter Right to left root node "))
    root.left.left.left = Node(input("Enter left to left to left root n
    root.left.left.right = Node(input("Enter right to left to left root
    printLevelOrder(root)
```

```
Enter root nodeA
Enter left to root nodeB
Enter Right to root nodeC
Enter left to Right to root nodeF
Enter left to left root nodeD
Enter Right to left root nodeE
Enter left to left to left root nodeG
Enter right to left to left root nodeH
A B C D E F G H
```

In [ ]:

```
1
```