Q.To print the right view of a binary tree, we can perform a modified level-order traversal (BFS) and print the rightmost node at each level. This way, we ensure that only the nodes visible from the right side are printed.

Here's the Python code to accomplish this:

```python

from collections import deque

class Node:

def \_\_init\_\_(self, value):

self.data = value

self.left = None

self.right = None

def print\_right\_view(root):

if not root:

return

queue = deque()

queue.append(root)

while queue:

# Get the number of nodes at the current level

level\_size = len(queue)

for i in range(level\_size):

node = queue.popleft()

# Print the rightmost node at the current level

if i == level\_size - 1:

print(node.data, end=' ')

# Add the left and right child of the current node to the queue

if node.left:

queue.append(node.left)

if node.right:

queue.append(node.right)

# Test case 1

root = Node(1)

root.left = Node(2)

root.right = Node(3)

root.left.left = Node(4)

root.left.right = Node(5)

root.right.left = Node(6)

root.right.right = Node(7)

root.right.right.right = Node(8)

print("Right view of the tree is:", end=' ')

print\_right\_view(root)

print()

# Test case 2

root = Node(1)

root.left = Node(8)

root.left.left = Node(7)

print("Right view of the tree is:", end=' ')

print\_right\_view(root)

print()

```

Output:

```

Right view of the tree is: 1 3 7 8

Right view of the tree is: 1 8 7

```

The code uses a queue to perform a level-order traversal of the binary tree. At each level, the rightmost node is printed. The left and right children of each node are added to the queue for further traversal.

Q. Given a Binary Tree, The task is to print the **bottom view** from left to right. A node **x** is there in output if x is the bottommost node at its horizontal distance. The horizontal distance of the left child of a node x is equal to a horizontal distance of x minus 1, and that of a right child is the horizontal distance of x plus 1.

**Examples:**

**Input:**

20

/     \\

8         22

/      \\         \\

5         3        25

/    \\

10       14

**Output:** 5, 10, 3, 14, 25.

**Input:**

20

/     \\

8         22

/      \\      /   \\

5         3   4     25

/    \\

10       14

**Output:**

5 10 4 14 25.

**Explanation:**

If there are multiple bottom-most nodes for a horizontal distance from the root, then print the later one in the level traversal.

**3 and 4** are both the bottom-most nodes at a horizontal distance of 0, we need to print 4.

Answers: To print the bottom view of a binary tree, we can perform a level-order traversal while keeping track of the horizontal distance of each node. We will store the horizontal distance along with each node in a queue. As we traverse the tree, we update the horizontal distance for each child node and store it in the queue.

Here's the modified code to print the bottom view of a binary tree:

```python

from collections import deque

class Node:

def \_\_init\_\_(self, value):

self.data = value

self.left = None

self.right = None

self.hd = 0 # Horizontal distance

def print\_bottom\_view(root):

if not root:

return

# Dictionary to store the bottom view nodes at each horizontal distance

bottom\_view = {}

queue = deque()

queue.append(root)

while queue:

node = queue.popleft()

hd = node.hd

# Update the bottom view for the current horizontal distance

bottom\_view[hd] = node.data

# Update the horizontal distance for the child nodes and add them to the queue

if node.left:

node.left.hd = hd - 1

queue.append(node.left)

if node.right:

node.right.hd = hd + 1

queue.append(node.right)

# Print the nodes in the bottom view from left to right

for key in sorted(bottom\_view.keys()):

print(bottom\_view[key], end=' ')

# Test case 1

root = Node(20)

root.left = Node(8)

root.right = Node(22)

root.left.left = Node(5)

root.left.right = Node(3)

root.right.right = Node(25)

root.left.right.left = Node(10)

root.left.right.right = Node(14)

print("Bottom view of the tree is:", end=' ')

print\_bottom\_view(root)

print()

# Test case 2

root = Node(20)

root.left = Node(8)

root.right = Node(22)

root.left.left = Node(5)

root.left.right = Node(3)

root.right.left = Node(4)

root.right.right = Node(25)

root.left.right.left = Node(10)

root.left.right.right = Node(14)

print("Bottom view of the tree is:", end=' ')

print\_bottom\_view(root)

print()

```

Output:

```

Bottom view of the tree is: 5 10 3 14 25

Bottom view of the tree is: 5 10 4 14 25

```

The code maintains a dictionary `bottom\_view` to store the bottom view nodes at each horizontal distance. After the level-order traversal, the nodes are printed from the dictionary in the sorted order of their horizontal distance. This ensures that the bottommost nodes for each horizontal distance are printed.