**PROGRAM CODE**

from collections import namedtuple

from sys import stdout

Node = namedtuple(*'Node'*, *'data, left, right'*)

tree = Node(9,Node(7,Node(6,Node(3, None, None),None),

Node(8, None, None)),

Node(17,

Node(15,

Node(10, None, None),

Node(16, None, None)),

None))

class **bst**:

def **printwithspace**(*self*,i):

stdout.write(*"%i "* % i)

def **preorder**(*self*,node):

if node is not None:

*self*.printwithspace(node.data)

*self*.preorder(node.left)

*self*.preorder(node.right)

def **inorder**(*self*,node):

if node is not None:

*self*.inorder(node.left)

*self*.printwithspace(node.data)

*self*.inorder(node.right)

def **postorder**(*self*,node):

if node is not None:

*self*.postorder(node.left)

*self*.postorder(node.right)

*self*.printwithspace(node.data)

def **levelorder**(*self*,node, more=None):

if node is not None:

if more is None:

more = []

more += [node.left, node.right]

*self*.printwithspace(node.data)

if more:

*self*.levelorder(more[0], more[1:])

Obj = bst()

stdout.write(*'Preorder: '*)

Obj.preorder(tree)

stdout.write(*'\nInorder: '*)

Obj.inorder(tree)

stdout.write(*'\nPostorder: '*)

Obj.postorder(tree)

stdout.write(*'\nBreadth First Search Traversal: '*)

Obj.levelorder(tree)

stdout.write(*'\n'*)