class Node():

def \_init\_(self):

self.left=None

self.right=None

self.data=None

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data)

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.left.right=Node()

tree.left.right.data=5

tree.right.left=Node()

tree.right.left.data=6

tree.right.right=Node()

tree.right.right.data=7

tree.postorder\_traversal(Node=tree)

class Node():

def \_init\_(self):

self.left=None

self.right=None

self.data=None

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data)

def Height(self,Node):

if Node is None:

return 0

else:

return max(self.Height(Node.left),self.Height(Node.right))+1

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.left.right=Node()

tree.left.right.data=5

tree.right.left=Node()

tree.right.left.data=6

tree.right.right=Node()

tree.right.right.data=7

tree.postorder\_traversal(Node=tree)

print(tree.Height(Node=tree))

class Node():

def \_init\_(self):

self.left=None

self.right=None

self.data=None

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data)

def count\_nodes(self,Node):

if Node is None:

return 0

return 1+self.count\_nodes(Node.left)+self.count\_nodes(Node.right)

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.left.right=Node()

tree.left.right.data=5

tree.right.left=Node()

tree.right.left.data=6

tree.right.right=Node()

tree.right.right.data=7

tree.postorder\_traversal(Node=tree)

print(tree.count\_nodes(Node=tree))