BINARY TREE:

class TreeNode:

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

self.data = data

self.left = None

self.right = None

class BinaryTree:

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

self.root = None

def insert(self, root, item):

if root is None:

return TreeNode(item)

if item < root.data:

root.left = self.insert(root.left, item)

else:

root.right = self.insert(root.right, item)

return root

def inorder\_traversal(self, root):

if root:

self.inorder\_traversal(root.left)

print(root.data, end=' ')

self.inorder\_traversal(root.right)

def preorder\_traversal(self, root):

if root:

print(root.data, end=' ')

self.preorder\_traversal(root.left)

self.preorder\_traversal(root.right)

def postorder\_traversal(self, root):

if root:

self.postorder\_traversal(root.left)

self.postorder\_traversal(root.right)

print(root.data, end=' ')

if \_\_name\_\_ == "\_\_main\_\_":

tree = BinaryTree()

n = int(input("Enter number of elements to insert: "))

print("Enter the elements:")

for \_ in range(n):

el = int(input())

tree.root = tree.insert(tree.root, el)

print("Inorder traversal:")

tree.inorder\_traversal(tree.root)

print("\nPreorder traversal:")

tree.preorder\_traversal(tree.root)

print("\nPostorder traversal:")

tree.postorder\_traversal(tree.root)

print()

OUTPUT:

Enter number of elements to insert: 5

Enter the elements:

20

40

30

10

70

Inorder traversal:

10 20 30 40 70

Preorder traversal:

20 10 40 30 70

Postorder traversal:

10 30 70 40 20