Program:

class Node:

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

self.key=key

self.left=None

self.right=None

def insert(root, key):

if root is None:

return Node(key)

if key < root.key:

root.left = insert(root.left, key)

elif key > root.key:

root.right = insert(root.right, key)

return root

def inorder\_traversal(root):

if root:

inorder\_traversal(root.left)

print(root.key, end=" ")

inorder\_traversal(root.right)

def find\_min(node):

current = node

while current.left:

current = current.left

return current

def delete\_node(root, key):

if root is None:

return root

if key < root.key:

root.left = delete\_node(root.left, key)

elif key > root.key:

root.right = delete\_node(root.right, key)

else:

# Node found

if root.left is None:

return root.right

elif root.right is None:

return root.left

# Both children exist: find in-order successor

succ = find\_min(root.right)

root.key = succ.key

root.right = delete\_node(root.right, succ.key)

return root

def display( node, level=0, prefix="Root: "):

if node is not None:

print(" " \* level + prefix + str(node.key))

display(node.left, level + 1, "L---- ")

display(node.right, level + 1, "R---- ")

# Example usage:

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

root = None

for value in [15, 10, 18, 4, 11, 16, 20, 13]:

root = insert(root, value)

print("In-order traversal:")

inorder\_traversal(root) # Should print sorted values

print("\n")

print("Display tree:")

display(root)

root = delete\_node(root, 10)

print("\nAfter deleting 10:")

inorder\_traversal(root)

print("\n")

display(root)

output:

In-order traversal:

4 10 11 13 15 16 18 20

Display tree:

Root: 15

L---- 10

L---- 4

R---- 11

R---- 13

R---- 18

L---- 16

R---- 20

After deleting 10:

4 11 13 15 16 18 20

Root: 15

L---- 11

L---- 4

R---- 13

R---- 18

L---- 16

R---- 20