**Practical No.6**

**Title :**

Construct an expression tree from the given prefix expression (eg. +--a\*bc/def) and traverse it using post order traversal (non recursive) and then delete the entire tree.

**Objective :**

* To construct an expression tree from the given prefix expression tree (+--a\*bc/def).

**Source Code :**

class TreeNode:

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

self.val = val

self.left = None

self.right = None

def construct\_expression\_tree(expression):

ops = set(['+', '-', '\*', '/'])

stack = []

for char in reversed(expression):

if char in ops:

node = TreeNode(char)

node.left, node.right = stack.pop(), stack.pop()

stack.append(node)

else:

stack.append(TreeNode(char))

return stack.pop()

def postorder\_traversal(root):

if not root: return []

result, stack = [], [root]

while stack:

node = stack.pop()

result.append(node.val)

if node.left: stack.append(node.left)

if node.right: stack.append(node.right)

return result[::-1]

def print\_tree(root, level=0):

if root:

print\_tree(root.right, level + 1)

print(' ' \* 4 \* level + '->', root.val)

print\_tree(root.left, level + 1)

def delete\_tree(root):

if root:

delete\_tree(root.left)

delete\_tree(root.right)

del root

prefix = input("Enter the prefix expression: ")

tree = construct\_expression\_tree(prefix)

print("\nExpression Tree:")

print\_tree(tree)

print("\nPostorder Traversal (non-recursive):", postorder\_traversal(tree))

delete\_tree(tree)

print("Tree deleted successfully.")

**Output :**

