Assignment no.: 8

Name : Kapare Swapnali Namdev

Class : SE

Div : A

Roll no : A- 65

Problem statement :

Construct an expression tree from the given prefix expression , e.g.,+--a*b c/d e f, traverse it using post-order traversal(non-recursive),and then delete the entire tree.

Input :

```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

# Utility function to check if a character is an operator
def is_operator(c):
    return c in "+-*/"

# Step 1: Construct Expression Tree from Prefix Expression
def construct_expression_tree(prefix_expr):
```

```
    stack = []

    # Traverse the prefix expression in reverse order
    for symbol in reversed(prefix_expr):
        if not is_operator(symbol):
            # Operand: create node and push to stack
            node = Node(symbol)
            stack.append(node)
        else:
            # Operator: pop two nodes, make them children
            node = Node(symbol)
            node.left = stack.pop()
            node.right = stack.pop()
            stack.append(node)

    # Final element in stack is the root of the tree
    return stack[0]

# Step 2: Non-recursive Post-order Traversal using two stacks
def post_order_non_recursive(root):
    if not root:
        return

    stack1 = [root]
    stack2 = []

    while stack1:
        node = stack1.pop()
```

```
        stack2.append(node)

        # Push left and right children to stack1
        if node.left:
            stack1.append(node.left)
        if node.right:
            stack1.append(node.right)

    # Print nodes in post-order
    while stack2:
        node = stack2.pop()
        print(node.value, end=' ')

# Step 3: Delete the entire tree
def delete_tree(node):
    if node is None:
        return
    delete_tree(node.left)
    delete_tree(node.right)
    # Explicitly delete references
    node.left = None
    node.right = None
    node.value = None

# ---------- Driver Code ----------

# Example prefix expression
prefix_expr = "+--a*bc/def"
```

```
# Step 1: Build Expression Tree
root = construct_expression_tree(prefix_expr)

# Step 2: Non-recursive Post-order Traversal
print("Post-order traversal (non-recursive):")
post_order_non_recursive(root)
print()

# Step 3: Delete the tree
delete_tree(root)
```

Output :

```
Post-order traversal (non-recursive):
a b c * - d e / - f +
```