BINARY TREE ADT

T SADAKOPA RAMAKRISHNAN | 3122225002109 | IT - B

Q1) Write a parser that takes an expression string in postfix notation (for eg, "ab+a*cd-e+/afg-*h+-) and constructs the corresponding expression tree. You may assume that only binary operators are used in the expression and all the identifiers are single characters only.

```
# -*- coding: utf-8 -*-
"""
This module provides the expression tree in which when gives
inorder expression as output. This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed Jun 21 2023

Revised on Wed Jul 01 2023

Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""



from linkedbinarytree import LinkedBinaryTree

class ExpressionTree(LinkedBinaryTree):
```

```
def __init__(self, item=None, Tleft=None, Tright=None):
    super().__init__(item, Tleft, Tright)

def construct(self, expr):
    S = []
    for ch in expr:
        if ch in '+*-/':
            rchild = S.pop()
            lchild = S.pop()
            S.append(ExpressionTree(ch, lchild, rchild))
        else:
            S.append(ExpressionTree(ch))
    return S.pop()

def inorder(self, pos):
    res = ""
    if pos is None:
        return None
    if pos.left is not None:
        res += self.inorder(pos.left)
    res += str(pos.item)
    if pos.right is not None:
        res += self.inorder(pos.right)
    return res

def __str__(self):
    return str(self.inorder(self._root))

def mirror(self, pos):
    if pos.left is not None:
        self.mirror(pos.left)
```

```
T1 = LinkedBinaryTree()
T2 = LinkedBinaryTree()
T3 = LinkedBinaryTree()
T4 = LinkedBinaryTree()
T1.addRoot(10)
T2.addRoot(20)
T3.addRoot(40)
T4.addRoot(10)
T4.addRight(50, T4.root())
T4.addLeft(T3.root().item, T4.root())
print("Preorder:", T4)

if __name__ == '__main__':
    #input is ab*c/e-
    e1 = ExpressionTree()
    e1.root = e1.construct("ab*c/e-")
    print(e1.root)

    e2 = ExpressionTree()
    print(e2.construct("ab+a*cd-e+/afg-*h+-"))
```

Q2) Given a binary tree, write a Python code to convert the binary tree into its Mirror tree. Mirror of a Binary Tree T is another Binary Tree M(T) with left and right children of all non-leaf nodes interchanged.

```
# -*- coding: utf-8 -*-
"""
This module provides the mirroring of tree code . This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may contain bugs.
The source code has followed good coding practices.
```

Your comments and suggestions are welcome.

Created on Wed Jun 21 2023

Revised on Wed Jul 01 2023

Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""

```
from linkedbinarytree import LinkedBinaryTree

class Tree(LinkedBinaryTree):
    def __init__(self, item=None, Tleft=None, Tright=None):
        super().__init__(item, Tleft, Tright)

    def mirror(self, pos):
        if pos is None:
            return
        else:
            pos.left, pos.right = pos.right, pos.left #swapping right and left
child
            self.mirror(pos.left) #swapping left subtree
            self.mirror(pos.right) #swapping right subtree

    def mirrorTree(self):
        return self.mirror(self._root)

T1 = Tree(10)
T2 = Tree(20, Tright=T1)
T3 = Tree(40, Tleft=T2)
T4 = Tree(10, T3)
```

```
T4.addRight(50, T4.root())

print("Original Tree (Preorder):", T4)
T4.mirrorTree()
print("Mirrored Tree (Preorder):", T4)
```