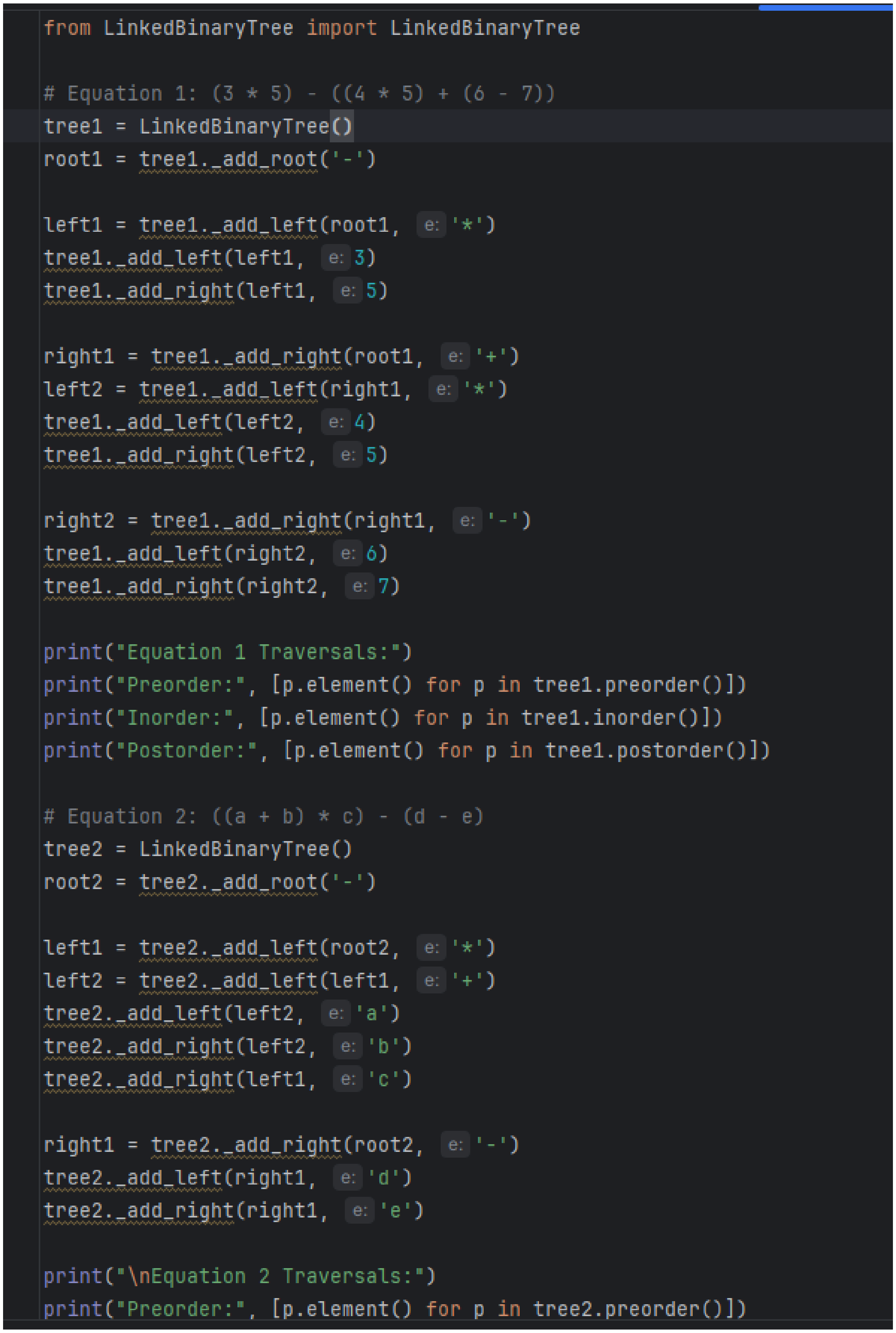
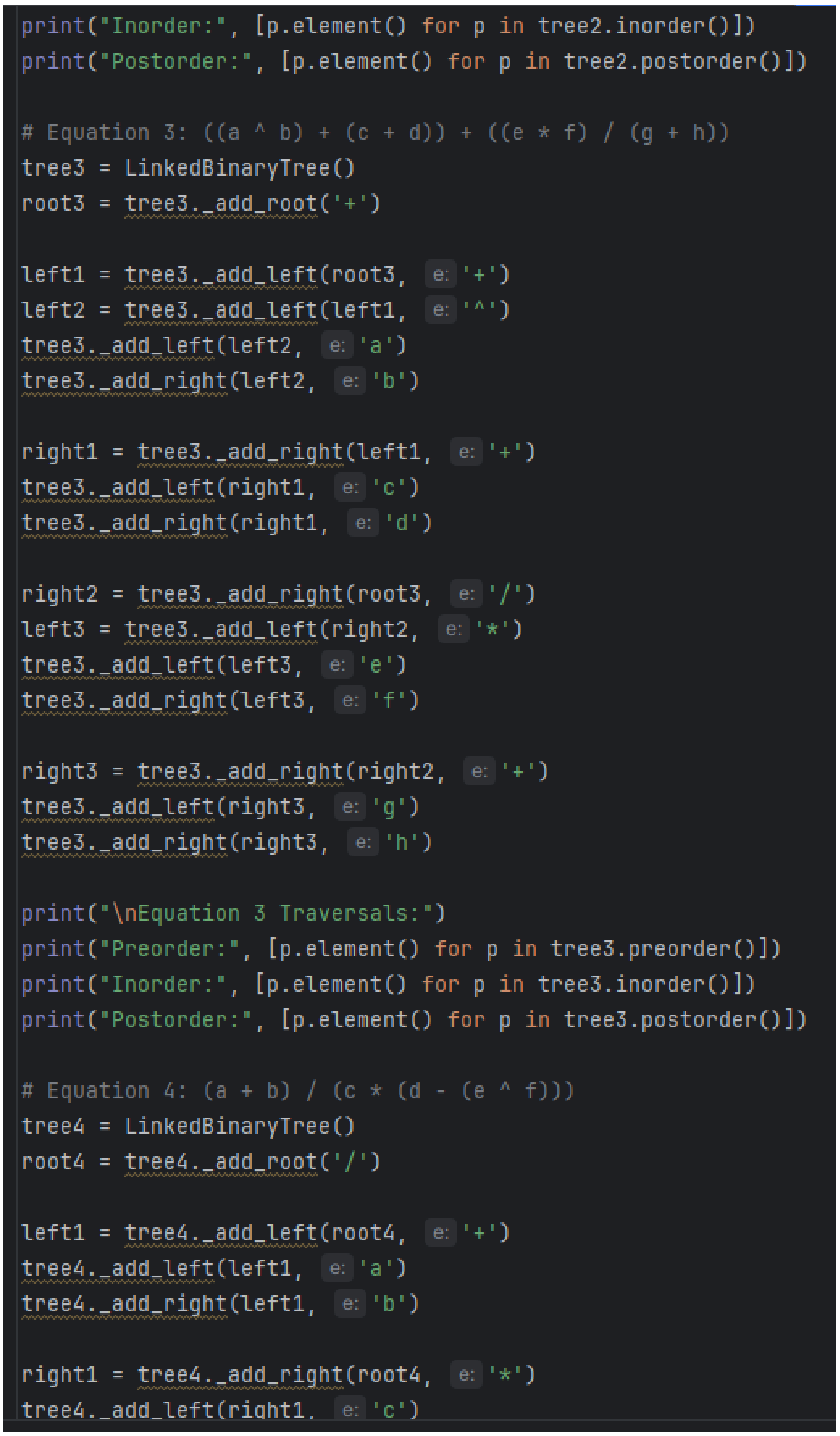
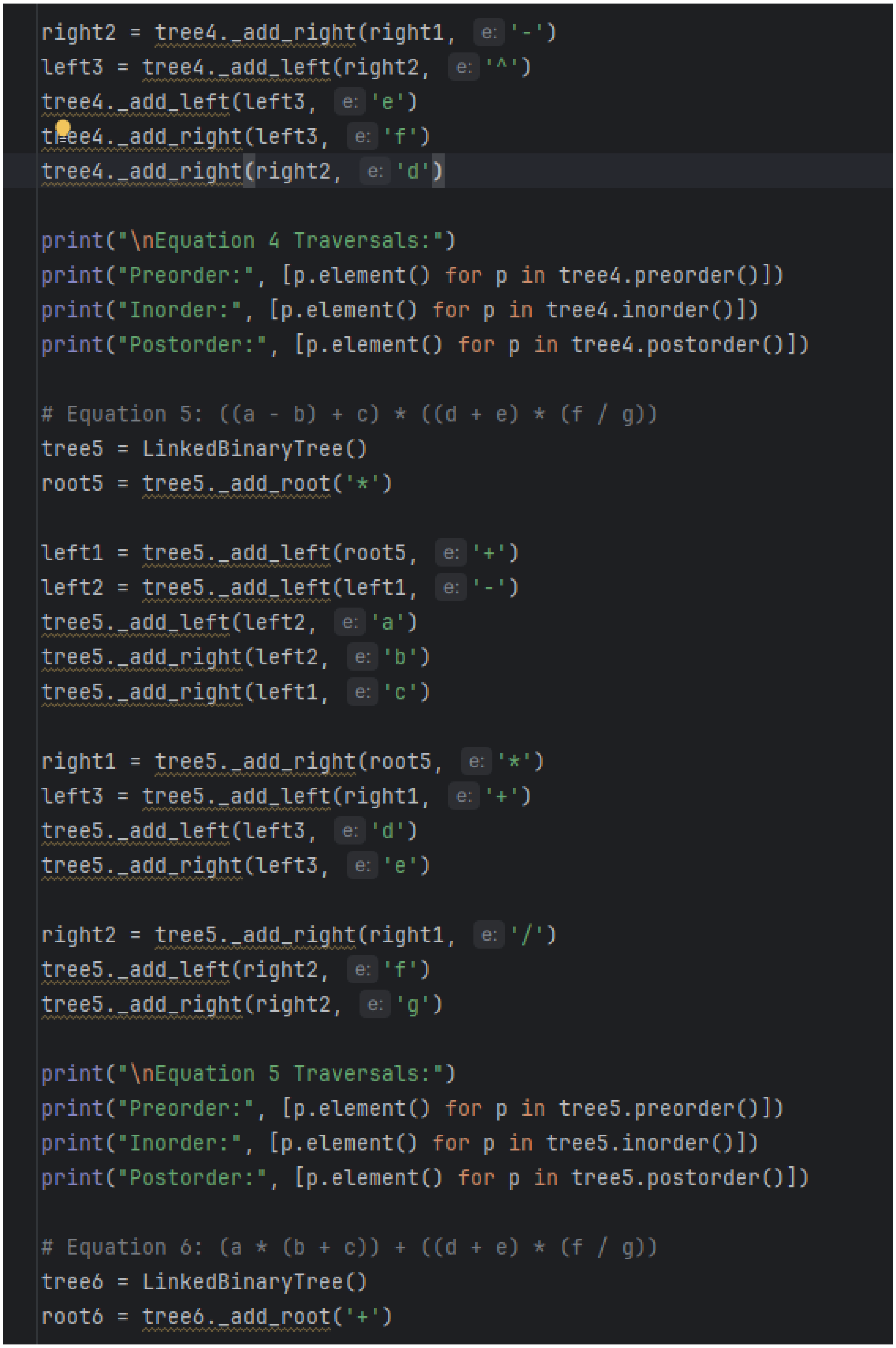


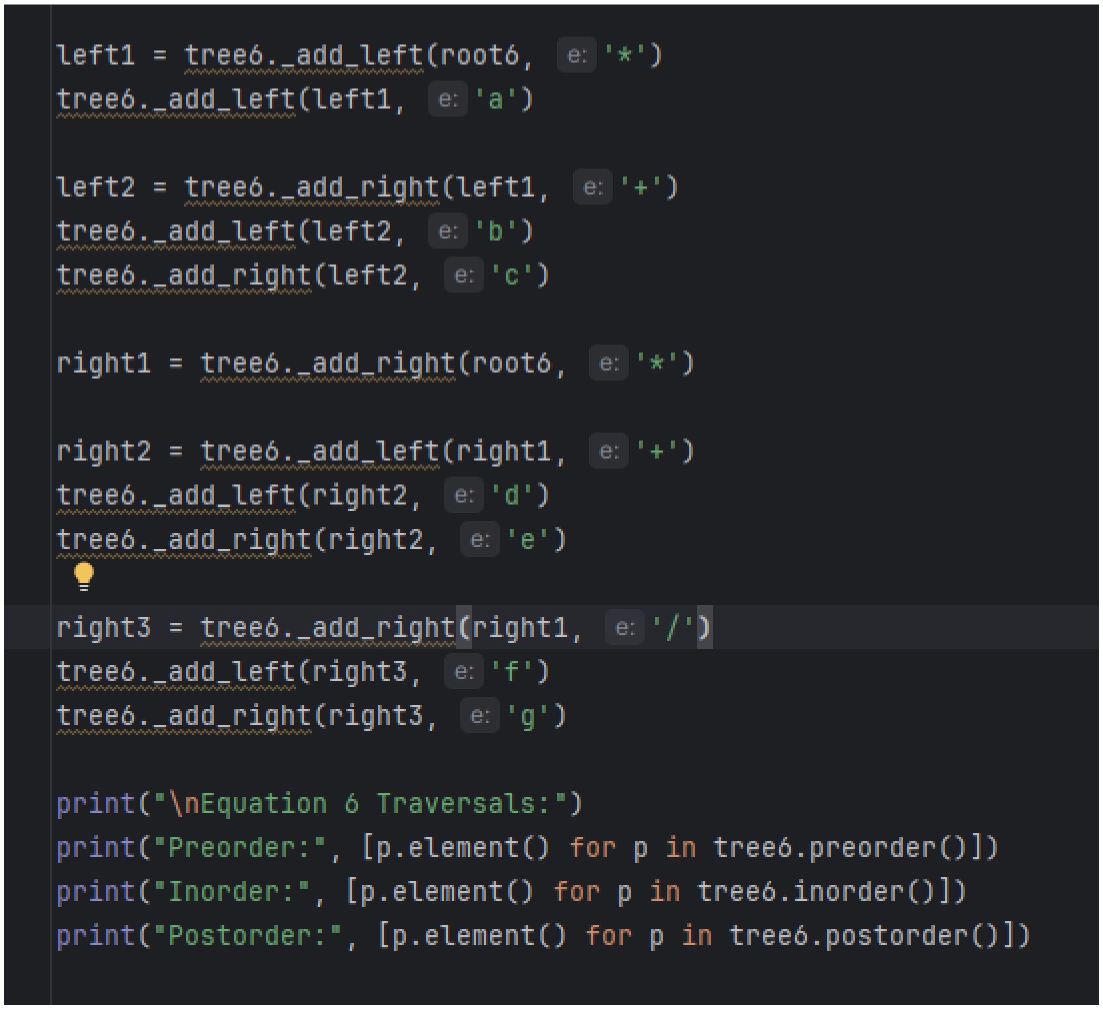
```python
from LinkedBinaryTree import LinkedBinaryTree

# Equation 1: (3 * 5) - ((4 * 5) + (6 - 7))
tree1 = LinkedBinaryTree()
root1 = tree1._add_root('-')

left1 = tree1._add_left(root1, e: '*')
tree1._add_left(left1, e: 3)
tree1._add_right(left1, e: 5)

right1 = tree1._add_right(root1, e: '+')
left2 = tree1._add_left(right1, e: '*')
tree1._add_left(left2, e: 4)
tree1._add_right(left2, e: 5)

right2 = tree1._add_right(right1, e: '-')
tree1._add_left(right2, e: 6)
tree1._add_right(right2, e: 7)

print("Equation 1 Traversals:")
print("Preorder:", [p.element() for p in tree1.preorder()])
print("Inorder:", [p.element() for p in tree1.inorder()])
print("Postorder:", [p.element() for p in tree1.postorder()])

# Equation 2: ((a + b) * c) - (d - e)
tree2 = LinkedBinaryTree()
root2 = tree2._add_root('-')

left1 = tree2._add_left(root2, e: '*')
left2 = tree2._add_left(left1, e: '+')
tree2._add_left(left2, e: 'a')
tree2._add_right(left2, e: 'b')
tree2._add_right(left1, e: 'c')

right1 = tree2._add_right(root2, e: '-')
tree2._add_left(right1, e: 'd')
tree2._add_right(right1, e: 'e')

print("\nEquation 2 Traversals:")
print("Preorder:", [p.element() for p in tree2.preorder()])
```

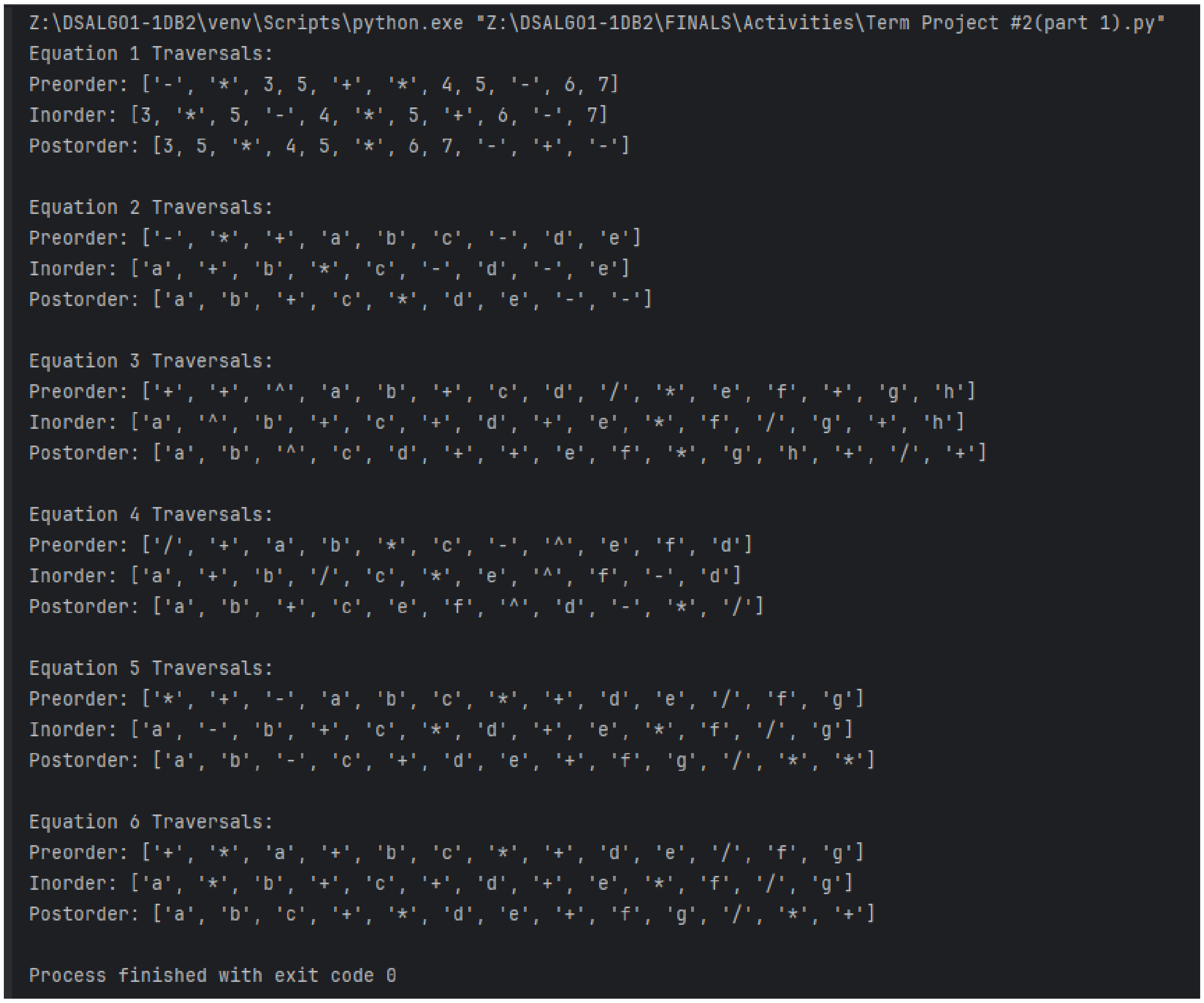


```python
print("Inorder:", [p.element() for p in tree2.inorder()])
print("Postorder:", [p.element() for p in tree2.postorder()])

# Equation 3: ((a ^ b) + (c + d)) + ((e * f) / (g + h))
tree3 = LinkedBinaryTree()
root3 = tree3._add_root('+')

left1 = tree3._add_left(root3, e: '+')
left2 = tree3._add_left(left1, e: '^')
tree3._add_left(left2, e: 'a')
tree3._add_right(left2, e: 'b')

right1 = tree3._add_right(left1, e: '+')
tree3._add_left(right1, e: 'c')
tree3._add_right(right1, e: 'd')

right2 = tree3._add_right(root3, e: '/')
left3 = tree3._add_left(right2, e: '*')
tree3._add_left(left3, e: 'e')
tree3._add_right(left3, e: 'f')

right3 = tree3._add_right(right2, e: '+')
tree3._add_left(right3, e: 'g')
tree3._add_right(right3, e: 'h')

print("\nEquation 3 Traversals:")
print("Preorder:", [p.element() for p in tree3.preorder()])
print("Inorder:", [p.element() for p in tree3.inorder()])
print("Postorder:", [p.element() for p in tree3.postorder()])

# Equation 4: (a + b) / (c * (d - (e ^ f)))
tree4 = LinkedBinaryTree()
root4 = tree4._add_root('/')

left1 = tree4._add_left(root4, e: '+')
tree4._add_left(left1, e: 'a')
tree4._add_right(left1, e: 'b')

right1 = tree4._add_right(root4, e: '*')
tree4._add_left(right1, e: 'c')
```

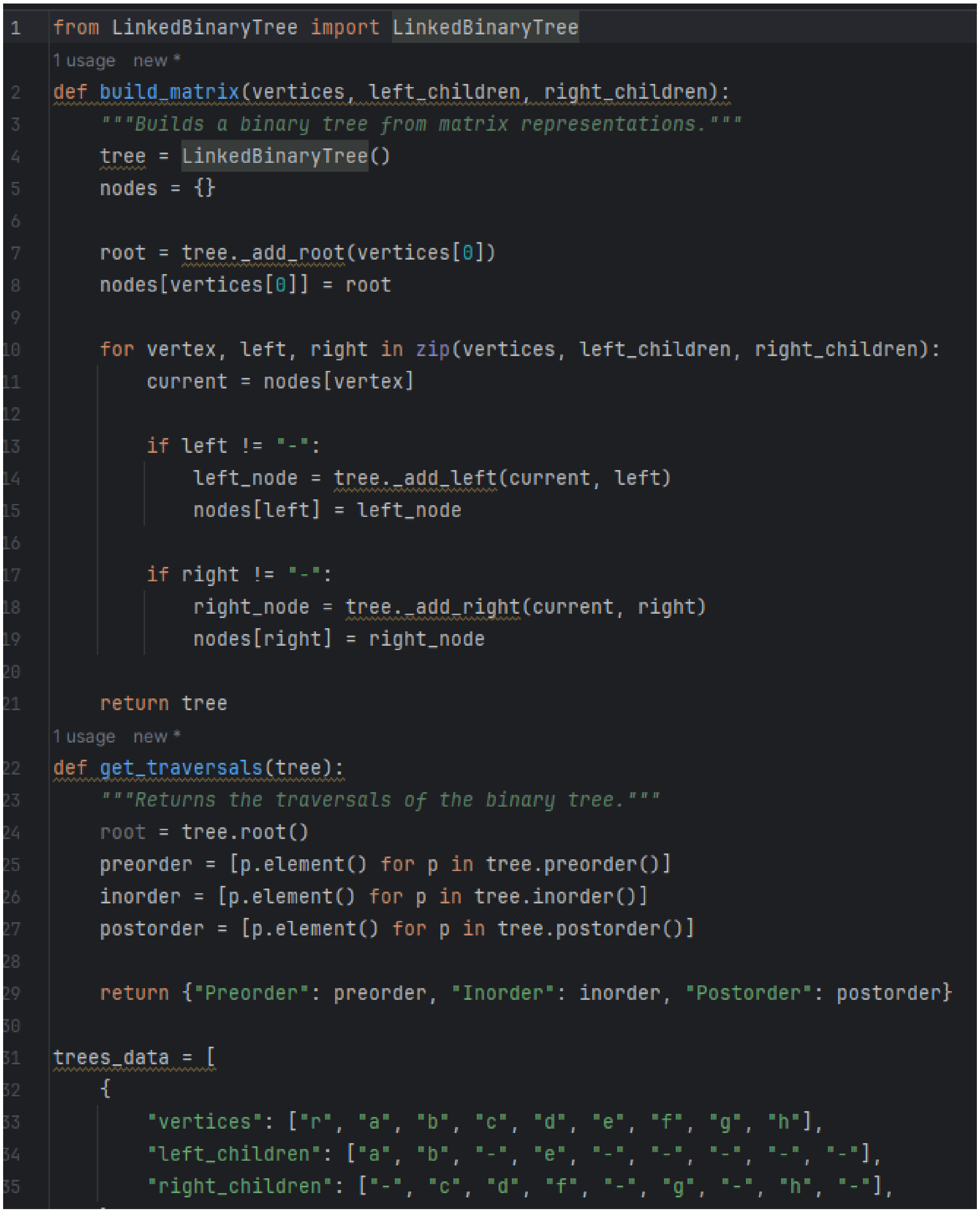


```python
right2 = tree4._add_right(right1, e: '-')
left3 = tree4._add_left(right2, e: '^')
tree4._add_left(left3, e: 'e')
tree4._add_right(left3, e: 'f')
tree4._add_right(right2, e: 'd')

print("\nEquation 4 Traversals:")
print("Preorder:", [p.element() for p in tree4.preorder()])
print("Inorder:", [p.element() for p in tree4.inorder()])
print("Postorder:", [p.element() for p in tree4.postorder()])

# Equation 5: ((a - b) + c) * ((d + e) * (f / g))
tree5 = LinkedBinaryTree()
root5 = tree5._add_root('*')

left1 = tree5._add_left(root5, e: '+')
left2 = tree5._add_left(left1, e: '-')
tree5._add_left(left2, e: 'a')
tree5._add_right(left2, e: 'b')
tree5._add_right(left1, e: 'c')

right1 = tree5._add_right(root5, e: '*')
left3 = tree5._add_left(right1, e: '+')
tree5._add_left(left3, e: 'd')
tree5._add_right(left3, e: 'e')

right2 = tree5._add_right(right1, e: '/')
tree5._add_left(right2, e: 'f')
tree5._add_right(right2, e: 'g')

print("\nEquation 5 Traversals:")
print("Preorder:", [p.element() for p in tree5.preorder()])
print("Inorder:", [p.element() for p in tree5.inorder()])
print("Postorder:", [p.element() for p in tree5.postorder()])

# Equation 6: (a * (b + c)) + ((d + e) * (f / g))
tree6 = LinkedBinaryTree()
root6 = tree6._add_root('+')
```

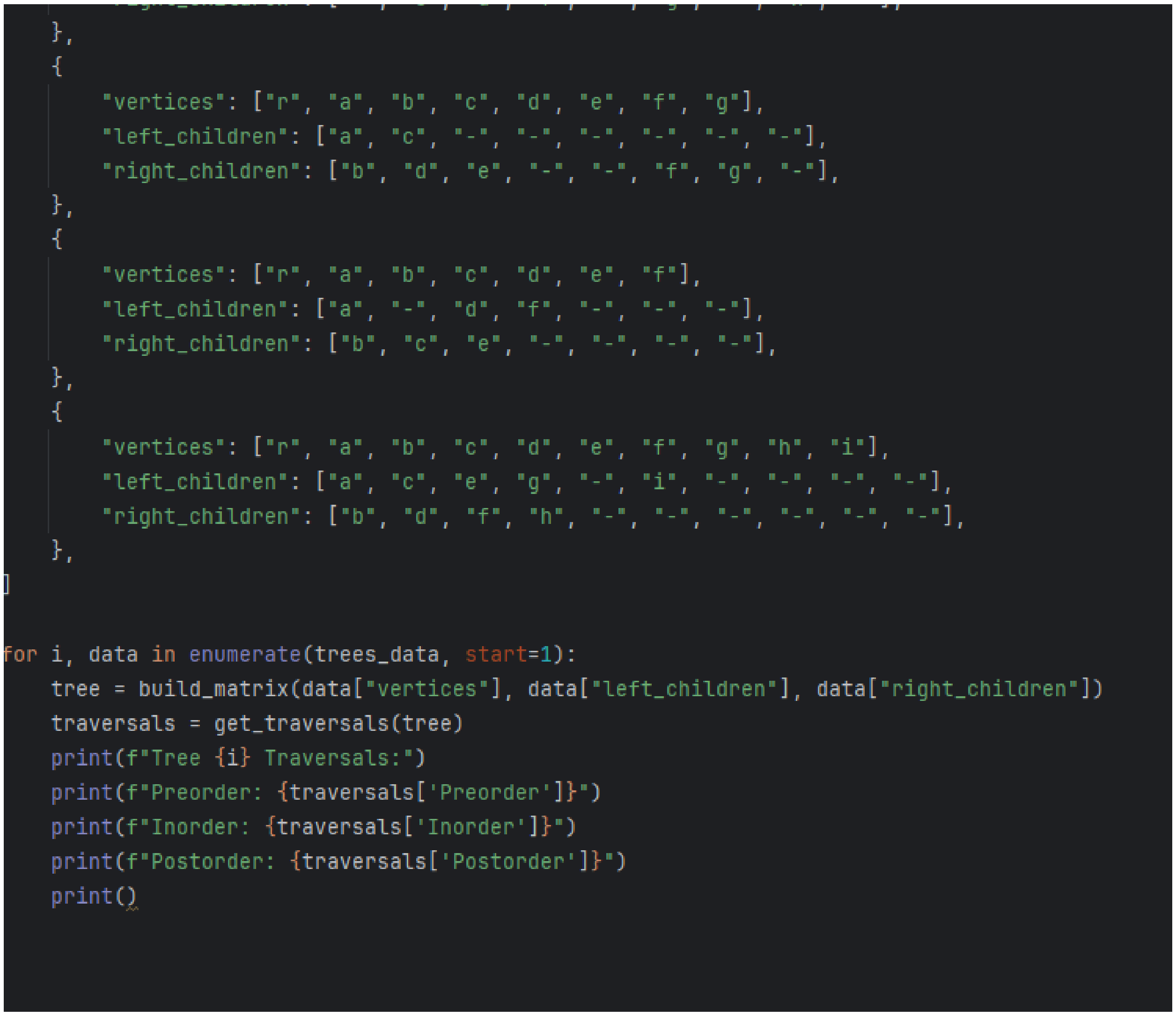

```
left1 = tree6._add_left(root6, e: '*')
tree6._add_left(left1, e: 'a')

left2 = tree6._add_right(left1, e: '+')
tree6._add_left(left2, e: 'b')
tree6._add_right(left2, e: 'c')

right1 = tree6._add_right(root6, e: '*')

right2 = tree6._add_left(right1, e: '+')
tree6._add_left(right2, e: 'd')
tree6._add_right(right2, e: 'e')

right3 = tree6._add_right(right1, e: '/')
tree6._add_left(right3, e: 'f')
tree6._add_right(right3, e: 'g')

print("\nEquation 6 Traversals:")
print("Preorder:", [p.element() for p in tree6.preorder()])
print("Inorder:", [p.element() for p in tree6.inorder()])
print("Postorder:", [p.element() for p in tree6.postorder()])
```



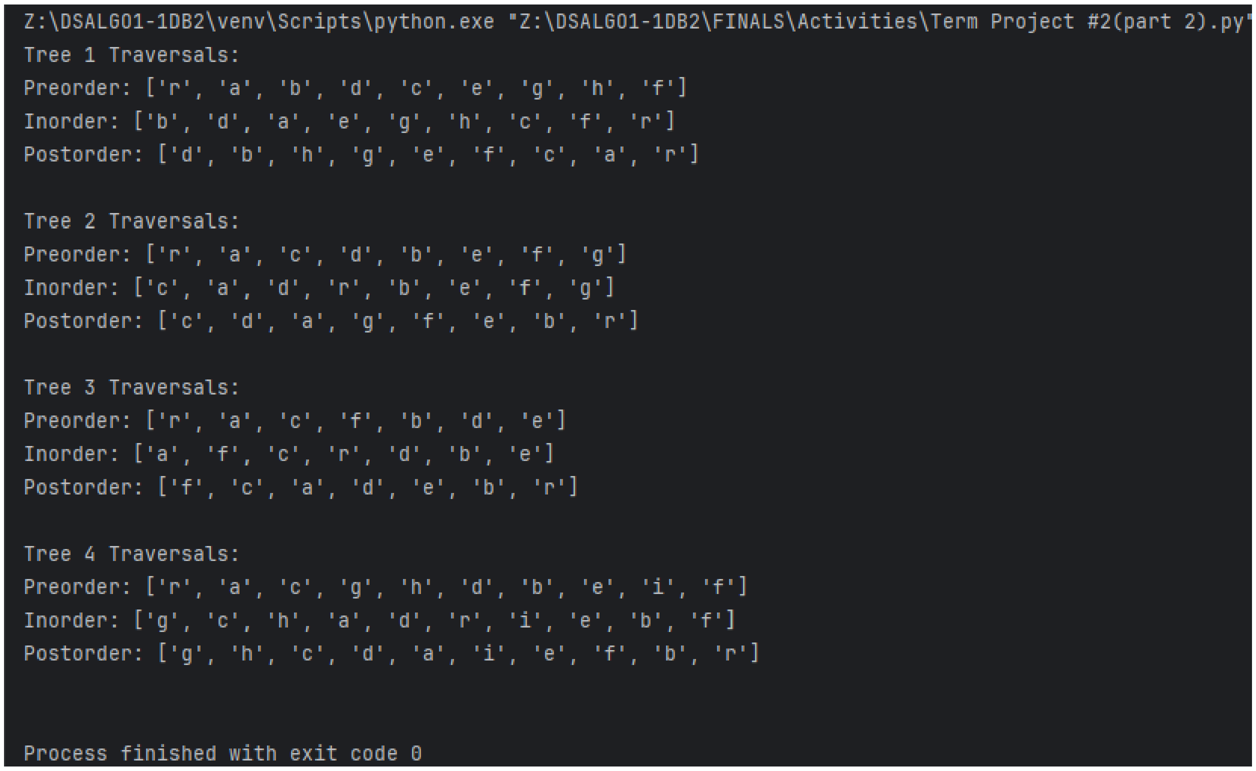
```
Z:\DSALGO1-1DB2\venv\Scripts\python.exe "Z:\DSALGO1-1DB2\FINALS\Activities\Term Project #2(part 1).py"
Equation 1 Traversals:
Preorder: ['-', '*', 3, 5, '+', '*', 4, 5, '-', 6, 7]
Inorder: [3, '*', 5, '-', 4, '*', 5, '+', 6, '-', 7]
Postorder: [3, 5, '*', 4, 5, '*', 6, 7, '-', '+', '-']

Equation 2 Traversals:
Preorder: ['-', '*', '+', 'a', 'b', 'c', '-', 'd', 'e']
Inorder: ['a', '+', 'b', '*', 'c', '-', 'd', '-', 'e']
Postorder: ['a', 'b', '+', 'c', '*', 'd', 'e', '-', '-']

Equation 3 Traversals:
Preorder: ['+', '+', '^', 'a', 'b', '+', 'c', 'd', '/', '*', 'e', 'f', '+', 'g', 'h']
Inorder: ['a', '^', 'b', '+', 'c', '+', 'd', '+', 'e', '*', 'f', '/', 'g', '+', 'h']
Postorder: ['a', 'b', '^', 'c', 'd', '+', '+', 'e', 'f', '*', 'g', 'h', '+', '/', '+']

Equation 4 Traversals:
Preorder: ['/', '+', 'a', 'b', '*', 'c', '-', '^', 'e', 'f', 'd']
Inorder: ['a', '+', 'b', '/', 'c', '*', 'e', '^', 'f', '-', 'd']
Postorder: ['a', 'b', '+', 'c', 'e', 'f', '^', 'd', '-', '*', '/']

Equation 5 Traversals:
Preorder: ['*', '+', '-', 'a', 'b', 'c', '*', '+', 'd', 'e', '/', 'f', 'g']
Inorder: ['a', '-', 'b', '+', 'c', '*', 'd', '+', 'e', '*', 'f', '/', 'g']
Postorder: ['a', 'b', '-', 'c', '+', 'd', 'e', '+', 'f', 'g', '/', '*', '*']

Equation 6 Traversals:
Preorder: ['+', '*', 'a', '+', 'b', 'c', '*', '+', 'd', 'e', '/', 'f', 'g']
Inorder: ['a', '*', 'b', '+', 'c', '+', 'd', '+', 'e', '*', 'f', '/', 'g']
Postorder: ['a', 'b', 'c', '+', '*', 'd', 'e', '+', 'f', 'g', '/', '*', '+']

Process finished with exit code 0
```



```python
from LinkedBinaryTree import LinkedBinaryTree
1 usage  new *
def build_matrix(vertices, left_children, right_children):
    """Builds a binary tree from matrix representations."""
    tree = LinkedBinaryTree()
    nodes = {}

    root = tree._add_root(vertices[0])
    nodes[vertices[0]] = root

    for vertex, left, right in zip(vertices, left_children, right_children):
        current = nodes[vertex]

        if left != "-":
            left_node = tree._add_left(current, left)
            nodes[left] = left_node

        if right != "-":
            right_node = tree._add_right(current, right)
            nodes[right] = right_node

    return tree
1 usage  new *
def get_traversals(tree):
    """Returns the traversals of the binary tree."""
    root = tree.root()
    preorder = [p.element() for p in tree.preorder()]
    inorder = [p.element() for p in tree.inorder()]
    postorder = [p.element() for p in tree.postorder()]

    return {"Preorder": preorder, "Inorder": inorder, "Postorder": postorder}

trees_data = [
    {
        "vertices": ["r", "a", "b", "c", "d", "e", "f", "g", "h"],
        "left_children": ["a", "b", "-", "e", "-", "-", "-", "-", "-"],
        "right_children": ["-", "c", "d", "f", "-", "g", "-", "h", "-"],
```



```python
    },
    {
        "vertices": ["r", "a", "b", "c", "d", "e", "f", "g"],
        "left_children": ["a", "c", "-", "-", "-", "-", "-", "-"],
        "right_children": ["b", "d", "e", "-", "-", "f", "g", "-"],
    },
    {
        "vertices": ["r", "a", "b", "c", "d", "e", "f"],
        "left_children": ["a", "-", "d", "f", "-", "-", "-"],
        "right_children": ["b", "c", "e", "-", "-", "-", "-"],
    },
    {
        "vertices": ["r", "a", "b", "c", "d", "e", "f", "g", "h", "i"],
        "left_children": ["a", "c", "e", "g", "-", "i", "-", "-", "-", "-"],
        "right_children": ["b", "d", "f", "h", "-", "-", "-", "-", "-", "-"],
    },
]

for i, data in enumerate(trees_data, start=1):
    tree = build_matrix(data["vertices"], data["left_children"], data["right_children"])
    traversals = get_traversals(tree)
    print(f"Tree {i} Traversals:")
    print(f"Preorder: {traversals['Preorder']}")
    print(f"Inorder: {traversals['Inorder']}")
    print(f"Postorder: {traversals['Postorder']}")
    print()
```



```
Z:\DSALGO1-1DB2\venv\Scripts\python.exe "Z:\DSALGO1-1DB2\FINALS\Activities\Term Project #2(part 2).py
Tree 1 Traversals:
Preorder: ['r', 'a', 'b', 'd', 'c', 'e', 'g', 'h', 'f']
Inorder: ['b', 'd', 'a', 'e', 'g', 'h', 'c', 'f', 'r']
Postorder: ['d', 'b', 'h', 'g', 'e', 'f', 'c', 'a', 'r']

Tree 2 Traversals:
Preorder: ['r', 'a', 'c', 'd', 'b', 'e', 'f', 'g']
Inorder: ['c', 'a', 'd', 'r', 'b', 'e', 'f', 'g']
Postorder: ['c', 'd', 'a', 'g', 'f', 'e', 'b', 'r']

Tree 3 Traversals:
Preorder: ['r', 'a', 'c', 'f', 'b', 'd', 'e']
Inorder: ['a', 'f', 'c', 'r', 'd', 'b', 'e']
Postorder: ['f', 'c', 'a', 'd', 'e', 'b', 'r']

Tree 4 Traversals:
Preorder: ['r', 'a', 'c', 'g', 'h', 'd', 'b', 'e', 'i', 'f']
Inorder: ['g', 'c', 'h', 'a', 'd', 'r', 'i', 'e', 'b', 'f']
Postorder: ['g', 'h', 'c', 'd', 'a', 'i', 'e', 'f', 'b', 'r']



Process finished with exit code 0
```