Gerardo, Charles Hanzel D.
Olivas, Carmie Sophia N.

DSALGO – IDC2
November 29, 2024

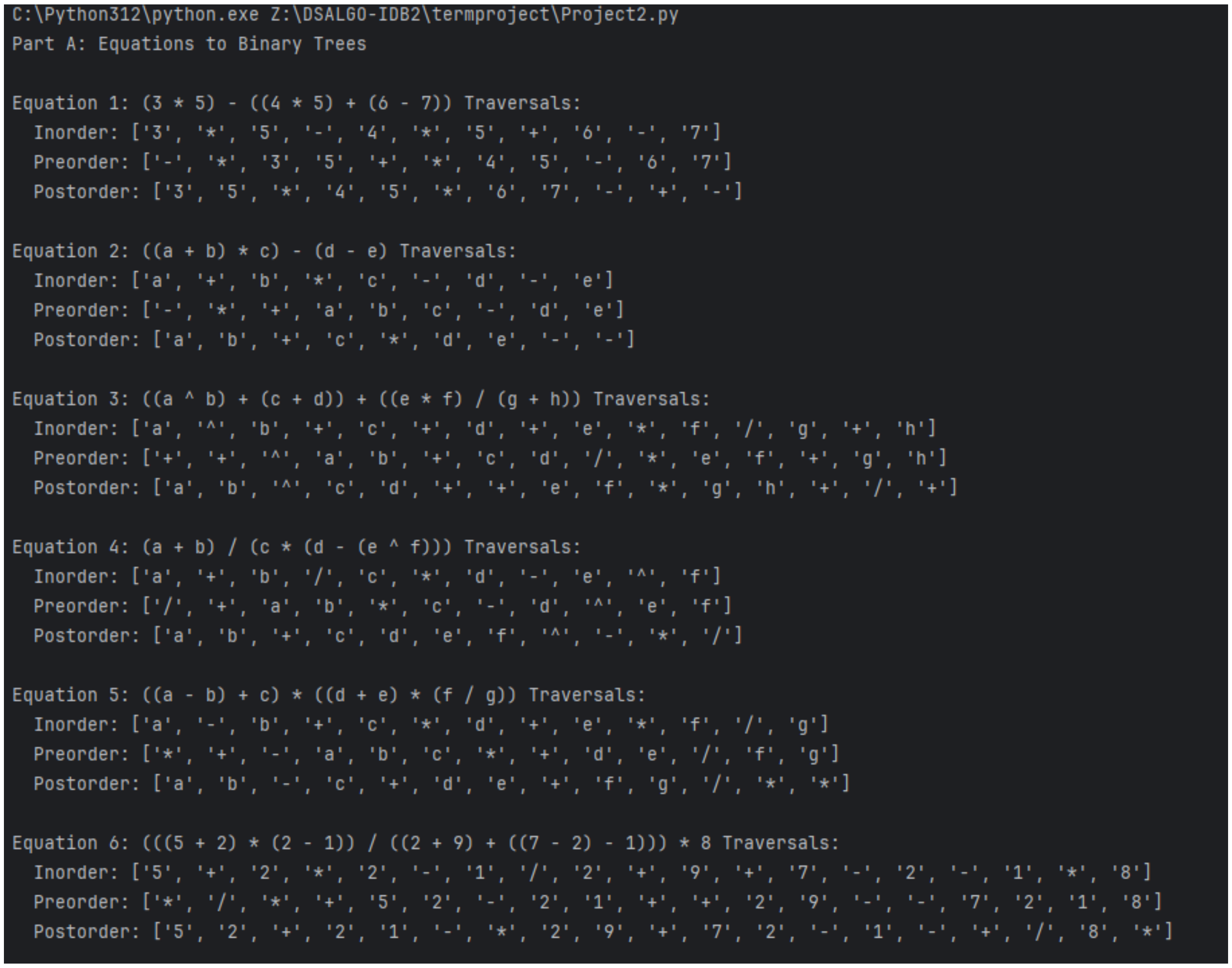
```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None


def inorder_traversal(node):
    return (inorder_traversal(node.left) if node.left else []) + [node.value]
+ \
        (inorder_traversal(node.right) if node.right else [])


def preorder_traversal(node):
    return [node.value] + (preorder_traversal(node.left) if node.left else
[]) + \
        (preorder_traversal(node.right) if node.right else [])


def postorder_traversal(node):
    return (postorder_traversal(node.left) if node.left else []) + \
        (postorder_traversal(node.right) if node.right else []) +
[node.value]


# Part A: Binary Trees for Equations
tree1 = Node('-')
tree1.left = Node('*')
tree1.left.left = Node('3')
tree1.left.right = Node('5')
tree1.right = Node('+')
tree1.right.left = Node('*')
tree1.right.left.left = Node('4')
tree1.right.left.right = Node('5')
tree1.right.right = Node('-')
tree1.right.right.left = Node('6')
tree1.right.right.right = Node('7')

tree2 = Node('-')
tree2.left = Node('*')
tree2.left.left = Node('+')
tree2.left.left.left = Node('a')
tree2.left.left.right = Node('b')
tree2.left.right = Node('c')
tree2.right = Node('-')
tree2.right.left = Node('d')
tree2.right.right = Node('e')

tree3 = Node('+')
tree3.left = Node('+')
tree3.left.left = Node('^')
tree3.left.left.left = Node('a')
tree3.left.left.right = Node('b')
tree3.left.right = Node('+')
tree3.left.right.left = Node('c')
tree3.left.right.right = Node('d')
```

```
tree3.right = Node('/')
tree3.right.left = Node('*')
tree3.right.left.left = Node('e')
tree3.right.left.right = Node('f')
tree3.right.right = Node('+')
tree3.right.right.left = Node('g')
tree3.right.right.right = Node('h')

tree4 = Node('/')
tree4.left = Node('+')
tree4.left.left = Node('a')
tree4.left.right = Node('b')
tree4.right = Node('*')
tree4.right.left = Node('c')
tree4.right.right = Node('-')
tree4.right.right.left = Node('d')
tree4.right.right.right = Node('^')
tree4.right.right.right.left = Node('e')
tree4.right.right.right.right = Node('f')

tree5 = Node('*')
tree5.left = Node('+')
tree5.left.left = Node('-')
tree5.left.left.left = Node('a')
tree5.left.left.right = Node('b')
tree5.left.right = Node('c')
tree5.right = Node('*')
tree5.right.left = Node('+')
tree5.right.left.left = Node('d')
tree5.right.left.right = Node('e')
tree5.right.right = Node('/')
tree5.right.right.left = Node('f')
tree5.right.right.right = Node('g')

tree6 = Node('*')
tree6.left = Node('/')
tree6.left.left = Node('*')
tree6.left.left.left = Node('+')
tree6.left.left.left.left = Node('5')
tree6.left.left.left.right = Node('2')
tree6.left.left.right = Node('-')
tree6.left.left.right.left = Node('2')
tree6.left.left.right.right = Node('1')
tree6.left.right = Node('+')
tree6.left.right.left = Node('+')
tree6.left.right.left.left = Node('2')
tree6.left.right.left.right = Node('9')
tree6.left.right.right = Node('-')
tree6.left.right.right.left = Node('-')
tree6.left.right.right.left.left = Node('7')
tree6.left.right.right.left.right = Node('2')
tree6.left.right.right.right = Node('1')
tree6.right = Node('8')


# Part B: Binary Trees from Matrix Representations
```

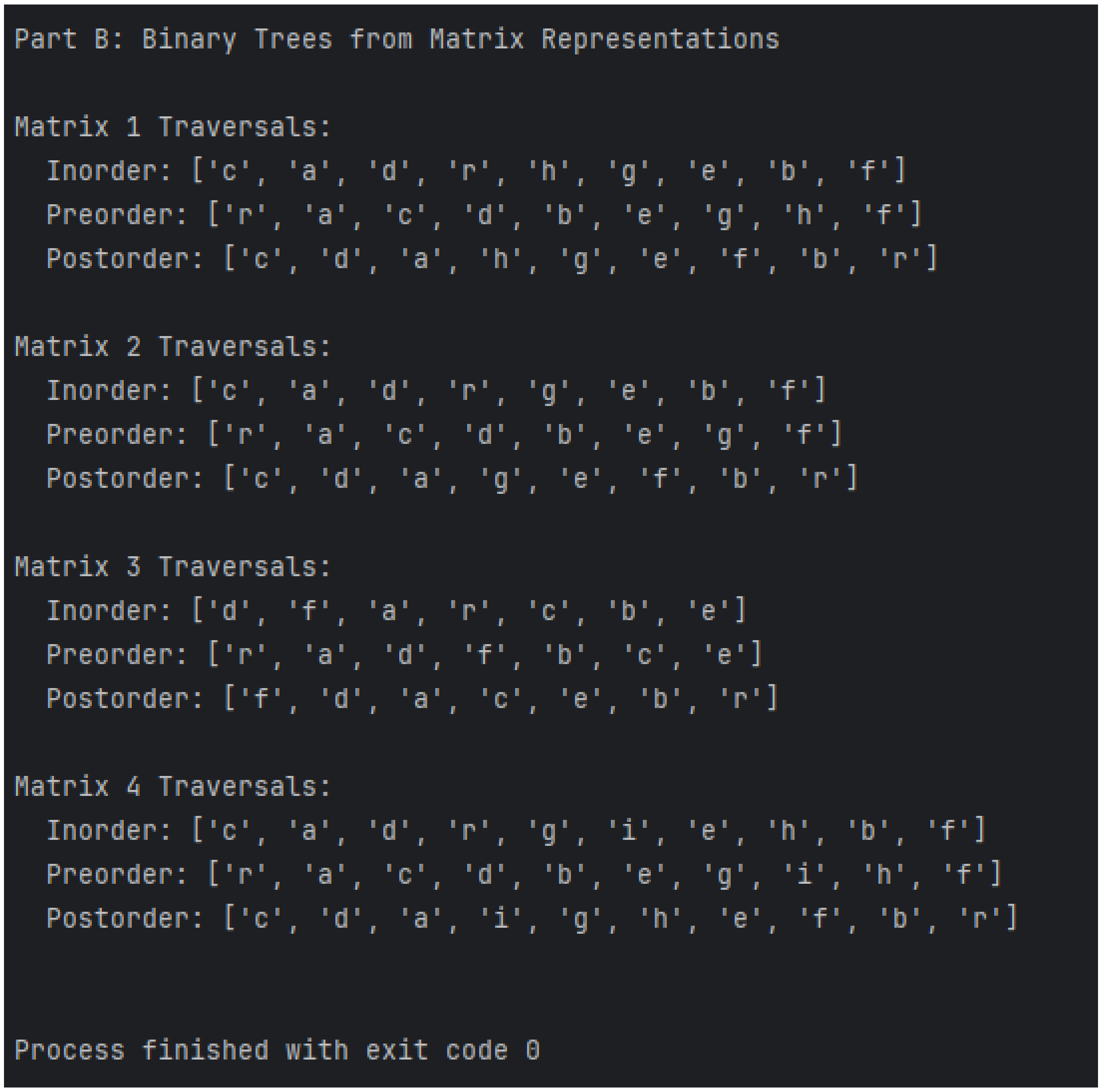
```python
def create_tree_matrix_1():
    nodes = {ch: Node(ch) for ch in 'rabcdefgh'}
    nodes['r'].left, nodes['r'].right = nodes['a'], nodes['b']
    nodes['a'].left, nodes['a'].right = nodes['c'], nodes['d']
    nodes['b'].left, nodes['b'].right = nodes['e'], nodes['f']
    nodes['e'].left = nodes['g']
    nodes['g'].left, nodes['g'].right = nodes['h'], None
    return nodes['r']


def create_tree_matrix_2():
    nodes = {ch: Node(ch) for ch in 'rabcdefg'}
    nodes['r'].left, nodes['r'].right = nodes['a'], nodes['b']
    nodes['a'].left, nodes['a'].right = nodes['c'], nodes['d']
    nodes['b'].left, nodes['b'].right = nodes['e'], nodes['f']
    nodes['e'].left = nodes['g']
    return nodes['r']


def create_tree_matrix_3():
    nodes = {ch: Node(ch) for ch in 'rabcdef'}
    nodes['r'].left, nodes['r'].right = nodes['a'], nodes['b']
    nodes['a'].left, nodes['a'].right = nodes['d'], None
    nodes['b'].left, nodes['b'].right = nodes['c'], nodes['e']
    nodes['d'].right = nodes['f']
    return nodes['r']


def create_tree_matrix_4():
    nodes = {ch: Node(ch) for ch in 'rabcdefghi'}
    nodes['r'].left, nodes['r'].right = nodes['a'], nodes['b']
    nodes['a'].left, nodes['a'].right = nodes['c'], nodes['d']
    nodes['b'].left, nodes['b'].right = nodes['e'], nodes['f']
    nodes['e'].left, nodes['e'].right = nodes['g'], nodes['h']
    nodes['g'].right = nodes['i']
    return nodes['r']


def print_traversals(tree, name):
    print(f"{name} Traversals:")
    print(f"  Inorder: {inorder_traversal(tree)}")
    print(f"  Preorder: {preorder_traversal(tree)}")
    print(f"  Postorder: {postorder_traversal(tree)}\n")


if __name__ == "__main__":
    print("Part A: Equations to Binary Trees\n")
    print_traversals(tree1, "Equation 1: (3 * 5) - ((4 * 5) + (6 - 7))")
    print_traversals(tree2, "Equation 2: ((a + b) * c) - (d - e)")
    print_traversals(tree3, "Equation 3: ((a ^ b) + (c + d)) + ((e * f) / (g
+ h))")
    print_traversals(tree4, "Equation 4: (a + b) / (c * (d - (e ^ f)))")
    print_traversals(tree5, "Equation 5: ((a - b) + c) * ((d + e) * (f /
g))")
    print_traversals(tree6, "Equation 6: (((5 + 2) * (2 - 1)) / ((2 + 9) +
((7 - 2) - 1))) * 8")
```

```
    print("Part B: Binary Trees from Matrix Representations\n")
    tree_matrix1 = create_tree_matrix_1()
    print_traversals(tree_matrix1, "Matrix 1")
    tree_matrix2 = create_tree_matrix_2()
    print_traversals(tree_matrix2, "Matrix 2")
    tree_matrix3 = create_tree_matrix_3()
    print_traversals(tree_matrix3, "Matrix 3")
    tree_matrix4 = create_tree_matrix_4()
    print_traversals(tree_matrix4, "Matrix 4")
```

Gerardo, Charles Hanzel D. DSALGO – IDC2
Olivas, Carmie Sophia N. November 29, 2024



```
C:\Python312\python.exe Z:\DSALGO-IDB2\termproject\Project2.py
Part A: Equations to Binary Trees

Equation 1: (3 * 5) - ((4 * 5) + (6 - 7)) Traversals:
  Inorder: ['3', '*', '5', '-', '4', '*', '5', '+', '6', '-', '7']
  Preorder: ['-', '*', '3', '5', '+', '*', '4', '5', '-', '6', '7']
  Postorder: ['3', '5', '*', '4', '5', '*', '6', '7', '-', '+', '-']

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

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

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

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

Equation 6: (((5 + 2) * (2 - 1)) / ((2 + 9) + ((7 - 2) - 1))) * 8 Traversals:
  Inorder: ['5', '+', '2', '*', '2', '-', '1', '/', '2', '+', '9', '+', '7', '-', '2', '-', '1', '*', '8']
  Preorder: ['*', '/', '*', '+', '5', '2', '-', '2', '1', '+', '+', '2', '9', '-', '-', '7', '2', '1', '8']
  Postorder: ['5', '2', '+', '2', '1', '-', '*', '2', '9', '+', '7', '2', '-', '1', '-', '+', '/', '8', '*']
```



```
Part B: Binary Trees from Matrix Representations

Matrix 1 Traversals:
  Inorder: ['c', 'a', 'd', 'r', 'h', 'g', 'e', 'b', 'f']
  Preorder: ['r', 'a', 'c', 'd', 'b', 'e', 'g', 'h', 'f']
  Postorder: ['c', 'd', 'a', 'h', 'g', 'e', 'f', 'b', 'r']

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

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

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


Process finished with exit code 0
```