CS 332 Programming Assignment P3: Tree Traversal

TIME ESTIMATE: 9 hours (3 hours per part)

DELIVERABLES: This Programming assignment has three parts. Each part requires that you create a programming solution to the same problem, tree traversal, using a different programming paradigm. Each part has its own deliverable and due date. Deliver all files by uploading to Canvas.

P3.1 requires that you implement a solution using the declarative paradigm (e.g. MatLab, C, Fortran, Python with no OO or functional elements).
P3.2 requires that you implement a solution using the object-oriented paradigm (e.g. Java, C++).
P3.3 requires that you implement a solution using the functional paradigm in Racket.

Only electronic documents submitted via Canvas are acceptable. Do not submit a hard copy of your assignment. Do not email your assignment to the course instructor or grader. Late assignments will be graded at half credit.

PROBLEM DESCRIPTION:

1. A tree is a graph, G = {N, E}, where
    a. Edges represent a parent-child relationship between nodes.
    b. There is a single node, the root, having no parent.
    c. All other nodes are involved in a parent-child relationship.
    d. Nodes have exactly one parent, except the root node.
    e. Nodes may have zero or more children.
2. Tree traversal refers to the action of visiting nodes in a tree. For this assignment, tree traversal requires that all nodes in a tree be visited. There are multiple forms of tree traversal.
    a. Depth First Post-Order Traversal requires that the deepest nodes be visited from left to right. See Figure 1 for an example.
    b. Breadth First Traversal requires that nodes be visited from left to right at each level before descending to the next level. See Figure 1 for an example.
3. Note that parents may have any number of children, and trees may not be balanced. This is implied in Figure 1.

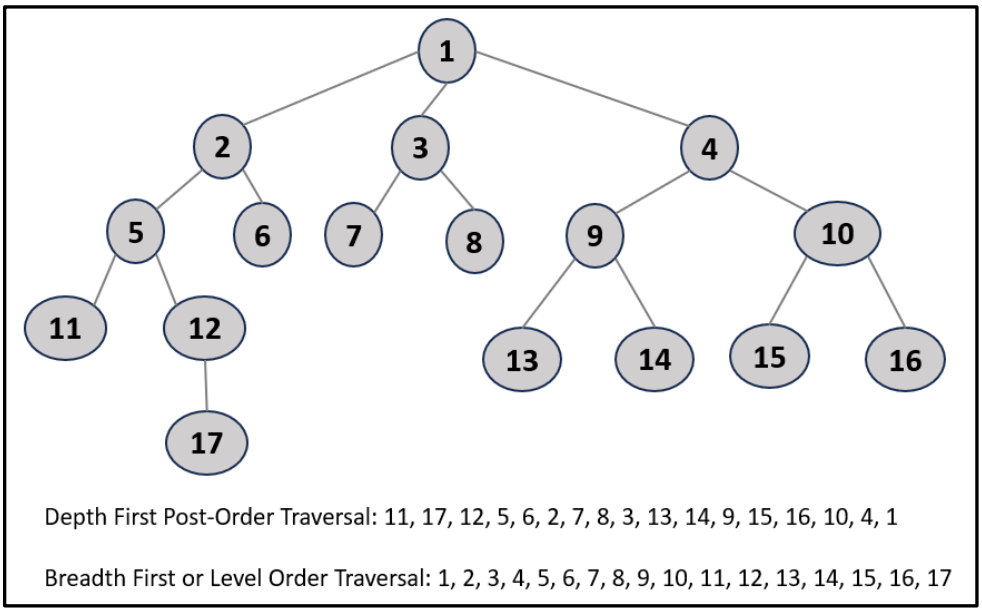

Figure 1: Example Tree Traversal (use for Test Case)

SOFTWARE REQUIREMENTS:
R1. The software solution shall be implemented three times, in the following programming paradigms: declarative (P3.1), object-oriented (P3.2), and functional (P3.3)
R2. The software shall perform the tests case in Figure 1 with no user input.
R3. For any given tree, T, the software shall return the sequence of nodes in Depth-First Preorder Traversal.
R4. For any given tree, T, the software shall return the sequence of nodes in Breadth-First Traversal
R6. The program shall return an empty list when given an empty tree.

TEST CASES: Use the tree and output provided in Figure 1 as a test case.

RUBRIC: Grades are distributed per the grading rubric in Table 1.

Table 1: Grading Rubric

Deliverable	P3.1	P3.2	P3.3
Program operates and produces output	10	10	10
Correct test case results	10	10	20
Correctness on other inputs	30	30	40
Totals	50	50	70