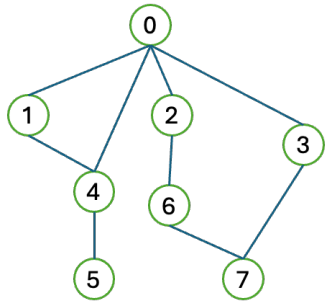
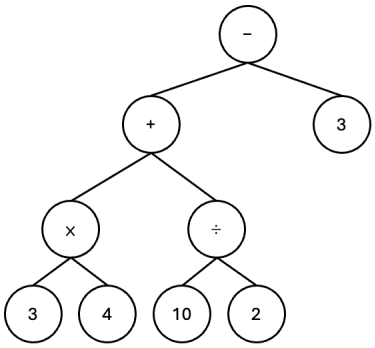
Course: Data Structures (CSE CS203A, 114-1)
Take-Home Quiz IV: Tree/Heap/Graph

**Due:** December 16, 2025, 17:00 (Room R1102)

**Important Notice**: You must print this take-home quiz sand **write your answers by hand with a pen**.

Student ID: Student Name:

| Q1 Figure | Q2 Figure |
|---|---|
|  |  |





Q1. (30 pts) Explain Breadth-First Search (BFS) on the graph and provide the BFS traversal order for the graph shown in Q1 Figure.

A1:

Breadth-First Search (BFS):

1. Uses a queue.
2. Starts from a selected node (usually the top-left or labeled root).
3. Visits nodes level by level (distance 1, then distance 2, etc.).
4. Ensures that all neighbors of the current node are visited before moving deeper.
5. Requires a visited set to avoid revisiting nodes.

Traversal order: 0, [any permutation of 1, 2, 3, 4], [children in the same parent order: 2->6, 3->7, 4->5]

All possible BFS traversals (24 total) are:

1. 0 → 1 → 2 → 3 → 4 → 6 → 7 → 5
2. 0 → 1 → 2 → 4 → 3 → 6 → 5 → 7
3. 0 → 1 → 3 → 2 → 4 → 7 → 6 → 5
4. 0 → 1 → 3 → 4 → 2 → 7 → 5 → 6
5. 0 → 1 → 4 → 2 → 3 → 5 → 6 → 7
6. 0 → 1 → 4 → 3 → 2 → 5 → 7 → 6
7. 0 → 2 → 1 → 3 → 4 → 6 → 7 → 5
8. 0 → 2 → 1 → 4 → 3 → 6 → 5 → 7
9. 0 → 2 → 3 → 1 → 4 → 6 → 7 → 5
10. 0 → 2 → 3 → 4 → 1 → 6 → 7 → 5

11. 0 → 2 → 4 → 1 → 3 → 6 → 5 → 7
12. 0 → 2 → 4 → 3 → 1 → 6 → 5 → 7
13. 0 → 3 → 1 → 2 → 4 → 7 → 6 → 5
14. 0 → 3 → 1 → 4 → 2 → 7 → 5 → 6
15. 0 → 3 → 2 → 1 → 4 → 7 → 6 → 5
16. 0 → 3 → 2 → 4 → 1 → 7 → 6 → 5
17. 0 → 3 → 4 → 1 → 2 → 7 → 5 → 6
18. 0 → 3 → 4 → 2 → 1 → 7 → 5 → 6
19. 0 → 4 → 1 → 2 → 3 → 5 → 6 → 7
20. 0 → 4 → 1 → 3 → 2 → 5 → 7 → 6
21. 0 → 4 → 2 → 1 → 3 → 5 → 6 → 7
22. 0 → 4 → 2 → 3 → 1 → 5 → 6 → 7
23. 0 → 4 → 3 → 1 → 2 → 5 → 7 → 6
24. 0 → 4 → 3 → 2 → 1 → 5 → 7 → 6

Q2. (30 pts) In tree traversal, one common method is inorder traversal. Please use inorder traversal to print the arithmetic expression represented by the expression tree in Q2 Figure, and then evaluate it to compute the final result.

A2:

Expression: 3 * 4 + 10 / 2 - 3

Result: 3 * 4 + 10 / 2 - 3 = 12 - 5 - 3 = 4

Q3. (40 pts) A binary tree is a fascinating data structure with many variations, including binary search trees, AVL trees, red-black trees, complete binary trees, and max/min heaps. These variations can be classified as shape-based (structural constraints) or criteria-based (rules such as ordering). Choose one shape-based tree and one criteria-based tree, and provide a brief description of each.

A3:

Shaped-based binary tree

1. Complete Binary Tree
   - Every level is fully filled except possibly the last (required)
   - Nodes in the last level are filled left to right (required)
   - Used in heap implementations (optional)
2. Full Binary Tree
   - Every node has 0 or 2 children (required)
3. Perfect Binary Tree
   - All internal nodes have 2 children and all leaves are at the same depth (required)

Criteria-based binary tree

1. Binary Search Tree (BST)
   - Left child < root < right child (required)
   - Searching, insertion, deletion average O(log n) (optional)
2. AVL Tree
   - Self-balancing BST (required)
   - Balance factor $\in \{-1, 0, +1\}$ (required)
   - Guarantees O(log n) operations(optional)
3. Red-Black Tree
   - Balanced BST with color rules (required)
   - Guarantees O(log n) height(optional)
4. Heap (Max/Min)
   - Parent $\geq$ (or $\leq$) children (required)
   - Complete shape + heap-order property (required)
   - Used in priority queues (optional)