1. Tree definition: A set of nodes
   1. one node is distinguished as _____ r
   2. Every node c except r has exactly one _____ node p connected by an edge. c is a _____ of p.
   3. A _____ path exists from r to each node.

   Where does it say about n-1 edges with n nodes?

2. More definitions

- Root: A distinguished node that has no parent
- L_____ node/e_____ node: A node with no children
- I_____ nodes: non-leaf nodes
- _____ of a node: the length of the path from the root to that node
- _____ of a node: the length of the path from that node to is deepest descendent (must be a leaf)

3. What is a binary tree?

4. How to implement a binary tree?

1) Use array:

   Let i be the index of a node.
   - What is the index of its left child?
   - What is the index of its right child?
   - What is the index of its parent?

2) Use a tree node with left/right references.

3) Pros/cons?

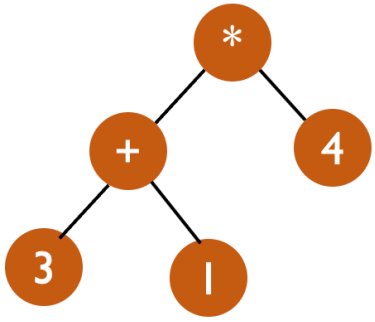
5. Implement inserting level order using tree node with left/right references.

```
// Level order insertion method using recursion
public void insertLevelOrder(int value) {
    TreeNode newNode = new TreeNode(value);

    // If the tree is empty, the new node becomes the root (Easy!)
    if (root == null) {
        root = newNode;
        return;
    }

    // Find the appropriate position and insert using recursion
    insert(root, newNode);
}
```

```
// Recursive method to insert a node level-by-level without queue or height
private void insert(TreeNode current, TreeNode newNode) {
```

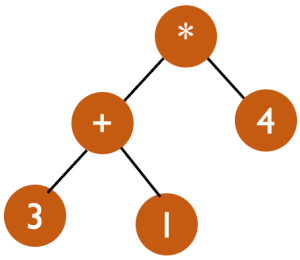
6. Binary tree traversal

- Pre-order
- In-order
- Post-order



7. Pre-order traversal
   Pseudo-code:

```
preorder(node)
    if(node == null) return;
    visit(node)
    preorder(node.left)//recurse on left subtree
    preorder(node.right) //recurse on right subtree
```



How to implement without recursion? What additional data structure to use?

```
// Iterative preorder traversal (root-left-right)
public void preorderIterative() {
    if (root == null) {
        return;
    }

    // Stack to store nodes for traversal
    Stack<TreeNode> stack = new Stack<>();
    //It's a pattern!
```

8. In-order traversal
   Pseudo-code:

```
inorder(node)
    if(node == null) return;
    inorder(node.left)//recurse on left subtree
    visit(node)
    inorder(node.right)//recurse on right subtree
```

How to implement without recursion? How many times should we visit each node?

In-order traversal without recursion (iterative approach)

```
// Iterative inorder traversal (left-root-right)
public void inorderIterative() {
    if (root == null) {
        return;
    }
    //note same pattern here!
    //need to somehow "mark" the second time visit and print
    Stack<TreeNode> stack = new Stack<>();
    stack.push(root);
    while (!stack.isEmpty()) {
        TreeNode node = stack.pop();
```

9. Post-order traversal
   Pseudo-code:

```
postorder(node)
    if(node == null) return;
    postorder(node.left)//recurse on left subtree
    postorder(node.right)//recurse on right subtree
    visit(node)
```

How to implement without recursion? How many times should we visit each node?

```
public void postorderIterative() { //Iterative postorder traversal(left-right-root)

    if (root == null) {
        return;
    }
    //note same pattern here!
    //need to somehow "mark" the second time visit and print
    Stack<TreeNode> stack = new Stack<>();
    stack.push(root);
    while (!stack.isEmpty()) {
        TreeNode node = stack.pop();
```

10. Level-order traversal
    Implementing without recursion is simpler here.
    Which traversal's code can we modify to level-order? (preorder, inorder, postorder)
    How to modify?

```
public void levelorderIterative() {// Iterative level order traversal
    if (root == null) {
        return;
    }
    // Queue to store nodes for traversal. Use the same pattern!
    Queue<TreeNode> queue = new LinkedList<>();
    queue.add(root);
    while (!queue.isEmpty()) {
```

11. What is the time complexity of binary tree traversal?