# 6. Trees – introduction.

## 1. The tree - abstract data type

A tree consists of a set of nodes and a set of edges (connecting pairs of nodes), satisfying the following properties:

- A tree has the property that there is exactly one path (no more, no less) between any pair of nodes, where a path is a connected sequence of edges.
- In a rooted tree, one distinguished node is called the root.

### Important expressions:

**node**: root, internal node (root included), leaf
**edge**: connecting pairs of nodes
**parent**
**ancestor**: parent, grandparent, grand-grandparent, etc
**depth of a node**: number of ancestors
**height of a tree**: maximum depth of any node
**child**
**descendant**: child, grandchild, grand-grandchild, etc.
**sibling**: two nodes that have the same parent
**path**: connected sequence of edges
**subtree**

## 2. The tree - traversal

In computer science, tree traversal (also known as tree search) is a form of graph traversal and refers to the process of visiting (checking and/or updating) each node in a tree exactly once. Such traversals are classified by the order in which the nodes are visited.

To traverse a tree using Preorder and Postorder, we can use two commonly used tree traversal methods. Both Preorder and Postorder are types of Depth First Search (DFS).

**Preorder** traversal visits the root node first and then traverses its left and right subtrees in that order. Here is the process:

1. Visit the root node.
2. Recursively traverse the left subtree.
3. Recursively traverse the right subtree.

**Postorder** traversal, on the other hand, visits the left and right subtrees first and then visits the root node. Here is the process:

1. Recursively traverse the left subtree.
2. Recursively traverse the right subtree.
3. Visit the root node.

## 3. Binary Tree

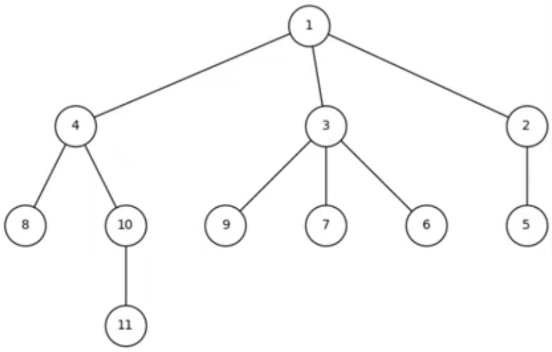
In computer science, a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child.

## 4. Expression tree:

An expression tree is a binary tree in which each internal node corresponds to the operator and each leaf node corresponds to the operand.

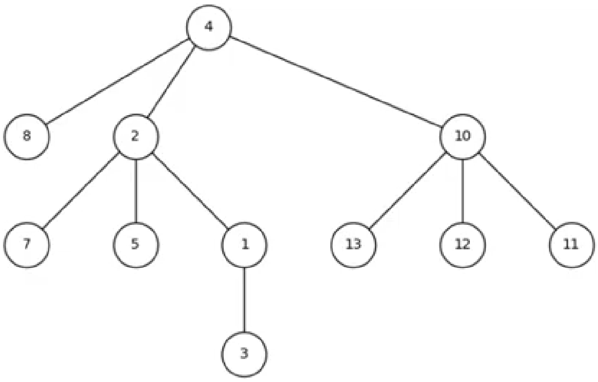
## 5. Exercises

1. Transform any tree into a hierarchical list

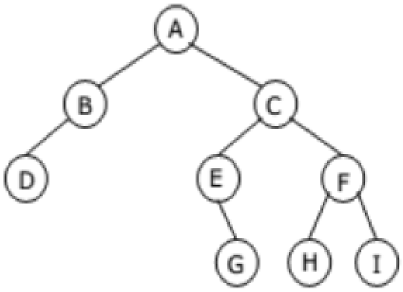



2. Transform a hierarchical list into a tree: **(1 (4 (8) (10 (11))) (3 (9) (7) (6)) (2(5)))**

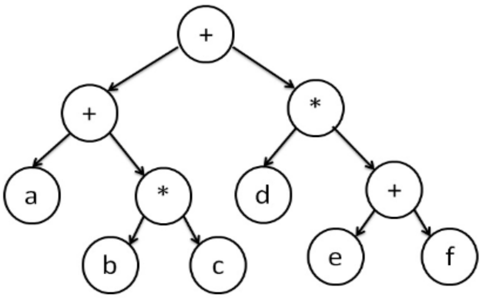
3. Preorder and postorder traversal of any tree.




4. Preorder and postorder traversal of the binary trees.




5. Prefix, infix, and postfix form of an expression tree.




6. Reconstruct an expression tree from a prefix or postfix form, and calculate its value if it contains only constants in its leaves (and no variables at all):

**Prefix expression: + - + 4 × 3 7 / 5 + 3 4 6**

**Postfix expression: 4 3 7 × + 5 3 4 + / - 6 +**