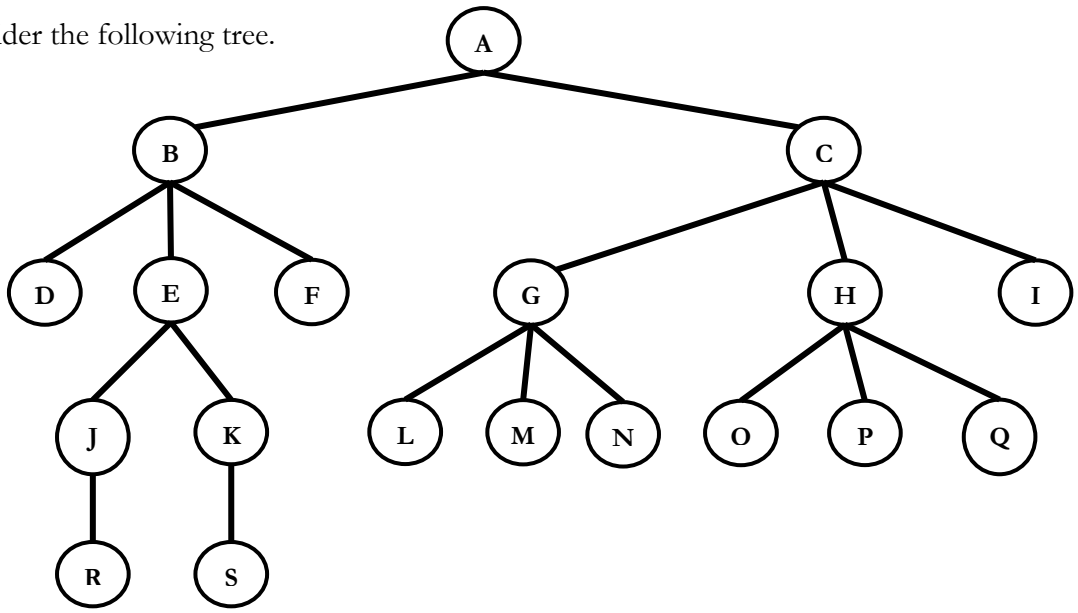
Exercise Series N°04 : Trees

Exercise IV.1 (Basic Tree Terminology)

Consider the following tree.



Q1) Give the values of these characteristics (root, degree of tree, number of nodes, external (leaves) and internal nodes height of tree, maximum depth of the tree) of the tree shown in the previous figure.

Q2) In the previous tree, write the pre-order, in-order and post-order traversals of each one.

Q3) Given the following properties of a tree, draw a tree that satisfies them:

- **a)** Degree of the tree: **3**
- **b)** Number of nodes: **14**
- **c)** Height of the tree: **3**
- **d)** Number of nodes with (**depth=2**) is : **6**

Exercise IV.2 (Properties of Binary Tree)

Prove these Theorems:

Q1) A perfect binary tree of height $\boldsymbol{h}$ has $\boldsymbol{2^{h+1} - 1}$ nodes.

Q2) The height of a perfect binary tree with $\boldsymbol{n}$ nodes is $\boldsymbol{h = log_2(n+1) - 1 = \mathcal{O}(log(n))}$.

Q3) A perfect binary tree of height $\boldsymbol{h}$ has $\boldsymbol{2^h}$ leaf nodes.

Exercise IV.3 (Generic Tree & Binary Tree)

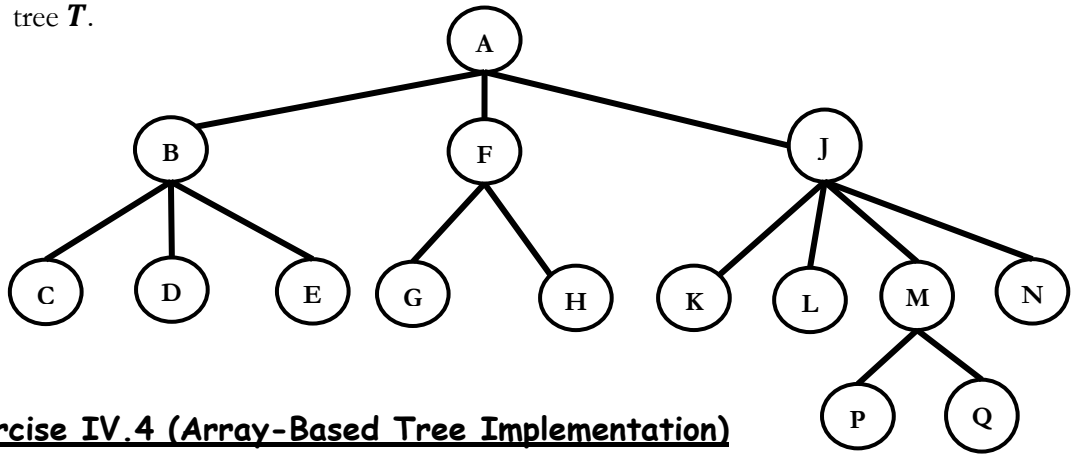
For the general tree $\boldsymbol{T}$ shown below:

Q1) Find the corresponding binary tree $\boldsymbol{T'}$.

Q2) Find the preorder traversal and the postorder traversal of $\boldsymbol{T}$.

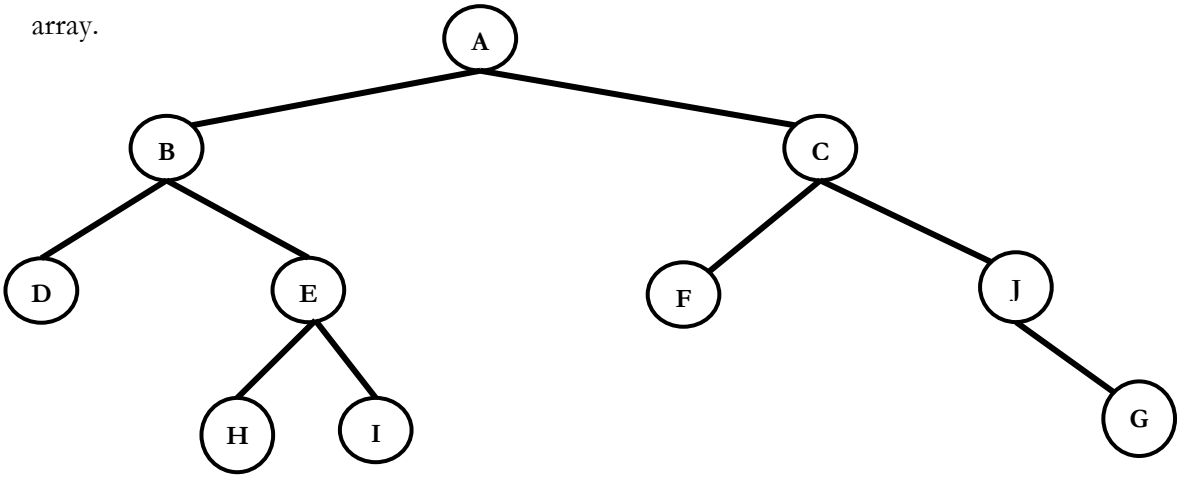
Q3) Find the preorder, inorder and postorder traversals of $\boldsymbol{T'}$.

Q4) Compare them with the preorder and postorder traversals obtained for T' with the general tree T.

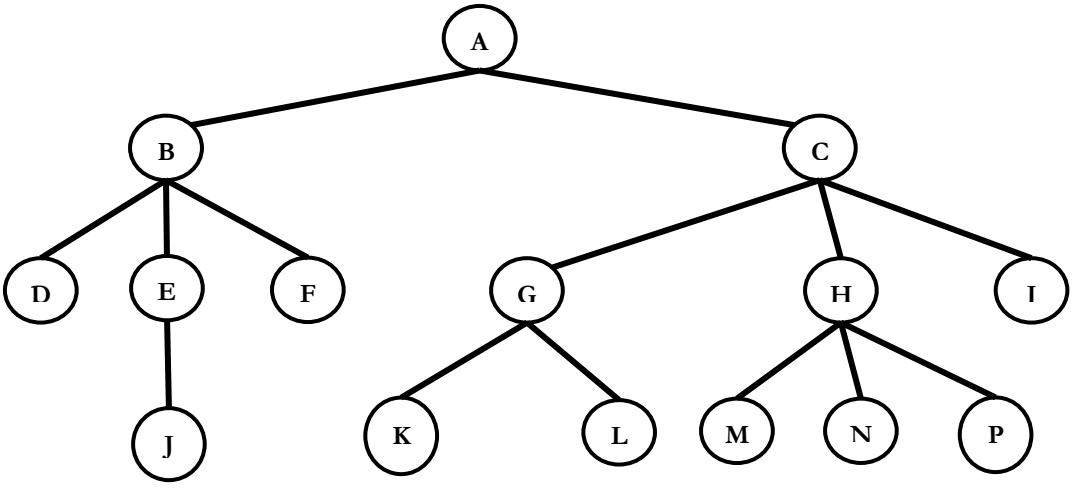


Exercise IV.4 (Array-Based Tree Implementation)

Q1) If we implement a binary tree using an array, we essentially flatten out the tree and store its nodes sequentially in memory, level by level. Represent the following binary tree using an array.



Q2) Suppose that we are using an array to represent a tree that can have three children: a left child, a middle child, and a right child. Represent the following tree using an array.



Exercise IV.5 (Binary Search Tree)

Q1) Draw a binary search tree containing keys **8, 27, 13, 15, 32, 20, 12, 50, 29, 11**, inserted in this order.

Q2) Then, add keys **14, 18, 30, 31**, in this order, and again draw the tree.

Q3) Then delete keys **29** and **27**, in this order, and again draw the tree.

Basic Binary Tree Functions

void preorder(BTREE BTree): A **preorder** traversal procedure.

void inorder(BTREE BTree) : An **inorder** traversal procedure.

void postorder(BTREE BTree): A **postorder** traversal procedure.

void level_order(BTREE BTree): A level-order traversal procedure.

Node* search_BT(BTREE BTree, **int** k): Searching for a key in a BST.

void insert_BST(BTREE &BTree, **int** k): Inserting a key into a BST.

void remove_BST(BTREE &BTree, **int** data): Removes the node from the BST.

Exercise IV.6 (Binary Tree Implementation)

A node in a binary tree is defined as follows:

```
struct BTNode {
    int data;
    BTNode *left;
    BTNode *right;
};
typedef BTNode *BTREE;
```

Q1) Write an **Init_BT()** function to initialize a binary tree.

Q2) Write a **Create_BTNode()** function to create a node in a binary tree**.**

Q3) Write a **IsEmpty_BT()** function to check if a binary tree is empty or not.

Q4) Write a **Size_BT_ Rec()** function for finding the size (number of node) of a binary tree **with recursion.**

Q5) Write a **Sum_BT()** function for finding the sum of all elements in binary tree.

Q6) Write a **Height()** function to find a height of any node in the binary tree.

What is the output when the **Height()** function is given the root of a binary tree as input?

Q7) Write a **Find_Level()** function to find the level of key in the Binary Tree.

Q8) Write a **Max_BT_Rec()** function for finding (searching) a maximum element in a binary tree **with recursion.**

Q9) Write a **Max_BT_It()** function for finding (searching) a maximum element in a binary tree **without recursion (iterative)**.

Q10) Write a **Count_Leaves()** function for finding the number of leaves in the binary tree