Trees and Graphs

(For the below tasks, you may want to create a binary tree manually and use the same tree for all of these tasks.)

NB: All the methods as well as the main method/tester statements must be written in one class. DO NOT write a different class for each method.

Compile all your codes and simulation picture in ONE PDF and submit it .

1. **RECURSIVELY** calculate the height of a tree.

2. **RECURSIVELY** calculate the level of a Node in a tree.

3. Print elements of all the Nodes of a tree using **Pre-order Traversal**.

4. Print elements of all the Nodes of a tree using **In-order Traversal**.

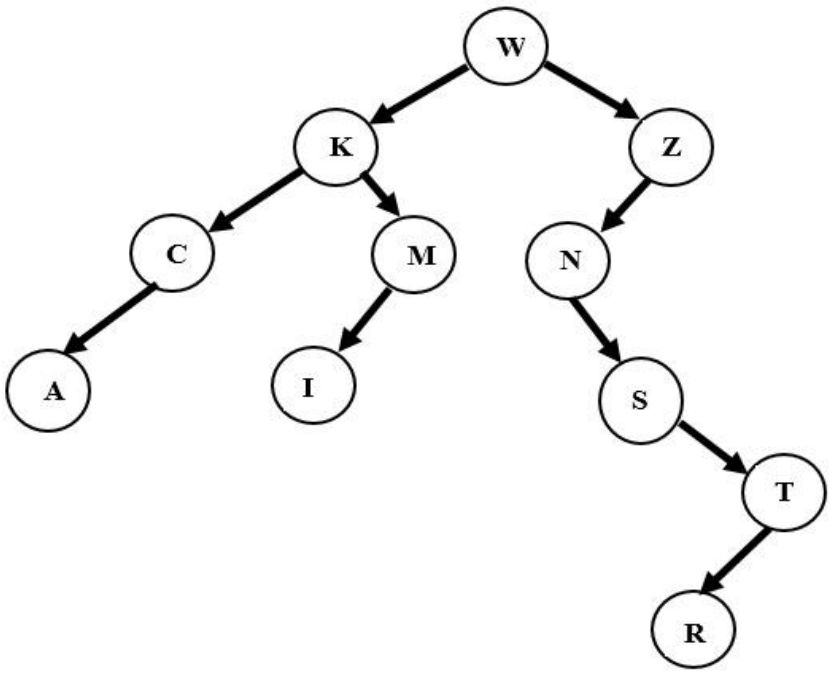
5. Print elements of all the Nodes of a tree using **Post-order Traversal**.

6. An adjacency matrix is given below:

	A	B	C	D	E	F	G
A	**0**	**1**	**0**	**1**	**1**	**0**	**0**
B	**0**	**0**	**0**	**0**	**0**	**0**	**1**
C	**0**	**1**	**0**	**0**	**0**	**0**	**0**
D	**0**	**0**	**1**	**0**	**0**	**0**	**1**
E	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F	**0**	**0**	**0**	**1**	**0**	**0**	**0**
G	**0**	**0**	**0**	**0**	**1**	**1**	**0**

a) Draw the equivalent graph.

7. Consider the following tree:

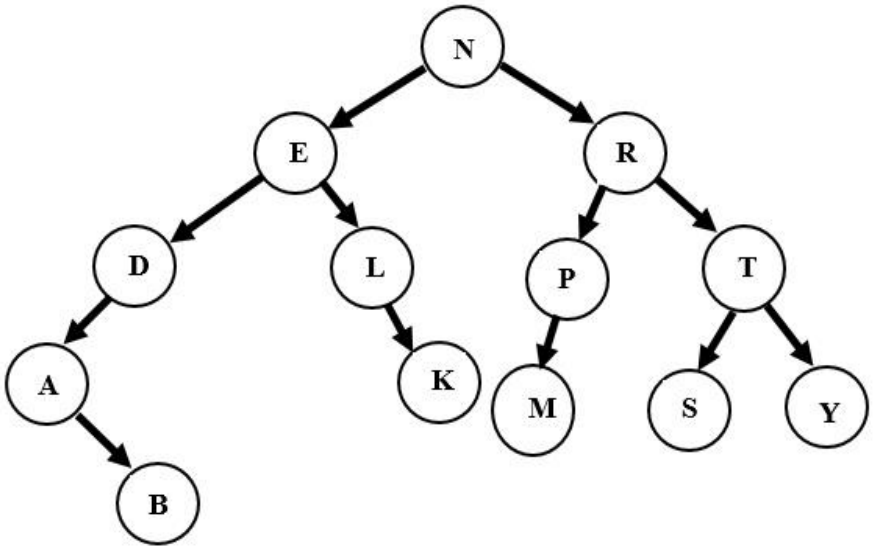


Write down the sequences for: Pre-order, in-order and Post-order Traversal. [2 + 2 +2 Marks]

a) Suppose we need to insert the sequence (3, 51, 6, 65, 17, 12, 1, 22, -3 and 15) into an initially empty binary search tree (of integers). Show the resultant Binary Search Tree.

[4 Marks]

b)



Do the following operations step by step on the above BST: [1 + 1 + 1 Marks]

a. Step 1: Remove node E with the help of its successor.
b. Step 2: Remove node N with the help of its successor.
c. Step 3: Remove node R with the help of its predecessor.