Tree

(For the below tasks, you may want to create a binary tree manually and use the same tree for all of these tasks. However, for task 6, you need two trees; hence, prepare another)

NB: All the methods(1-9) as well as the main method must be written in one class. DO NOT write a different class for each method.

Submit the single java file only DO NOT ZIP 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. Write a method which will evaluate whether two trees are **exactly same** or **not**.

7. Write a method which will return a **copy (new tree) of a given tree**.

8. Write a method that RECURSIVELY searches and finds an integer from a given binary search tree (BST).

9. Write a method that RECURSIVELY prints all the elements of a binary search tree in a sorted order.

10. 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.
b) Simulate BFS algorithm on the output of a.
c) Simulate DFS algorithm on the output of a.

See the pages 94~98 of “all-notes.pdf”