Data Structure Project

I. Submission Instructions:

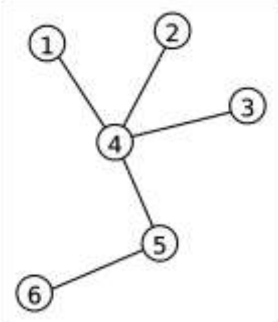
1- Submit your project as **StudentID.c** file

2- Add name and student ID as a comment in the first two lines

```
            //Name
            //StudentID
            #include<iostream>
            //Your code
```

II. Basic Definitions

Adjacency Matrix: is a matrix the represent a graph.

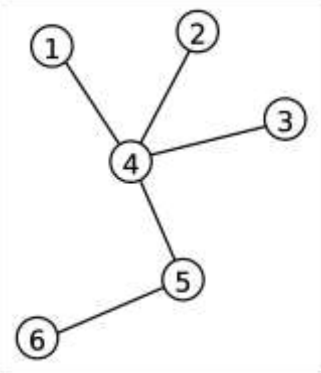


Graph

$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

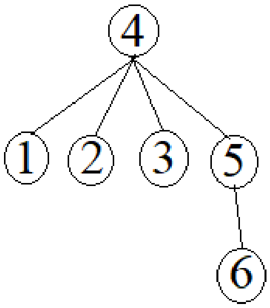
Adjacency Matrix

Unrooted tree: is a connected undirected graph with no simple circuits (cycles), no multiple edges and no loops.



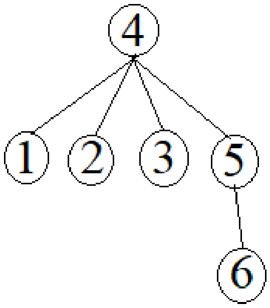
Unrooted Tree

Rooted tree: is a tree where one vertex designated as root and every edge directed away.



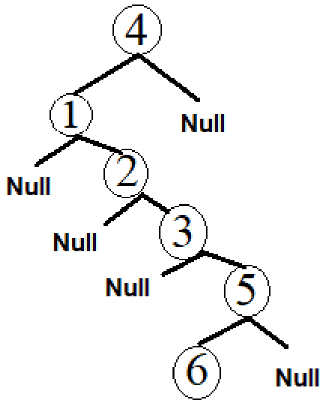
Rooted Tree with 4 as the root.

m-ary tree: is a tree where a parent can have up-to m children.

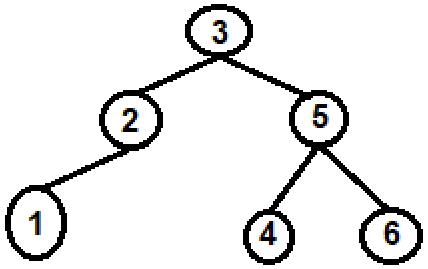


m-ary tree with m=4 (maximum number of children)

Binary tree: is a m-ary tree with m=2.

 4 1 Null Null 2 Null 3 Null 5 6 Null	Converting the above m-ary tree with the rule (in the lecture) Left=first child, Right = next brother This tree is not balanced

Balanced Binary search tree (AVL tree): is a binary search tree where for each node, the height of the left and right subtrees can differ at most 1.



Binary AVL search tree is the order of insertion: 1,2,3,6,5,4

III. Project

Given an adjacency matrix

1- Check if the adjacency matrix represents a m-ary unrooted tree m>=2. Otherwise terminate.
2- Allow the user to choose a node as the root.
3- Convert the m-ary tree in the adjacency matrix to a binary tree in a binary tree data structure taking the node chosen by the user as the root.
4- If the tree is not balanced, traverse the tree inorder and insert the traversed nodes in a Balanced Binary search tree (AVL tree).
5- Traverse the final AVL tree in preorder, inorder and postorder fashions.

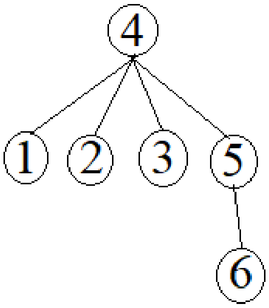
IV. Example:

1-Given the adjacency matrix

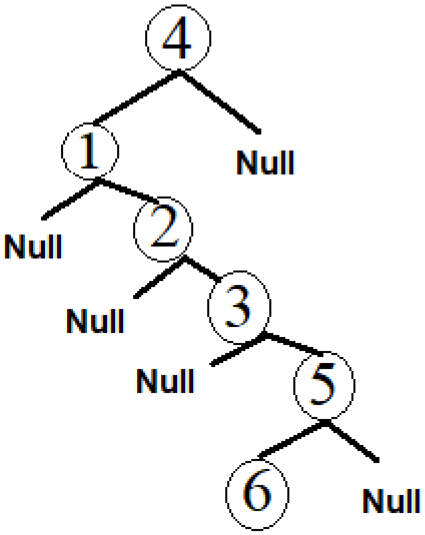
$$\begin{array}{c|cccccc|} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

2-Check if it represent a tree. no simple circuits (cycles), no multiple edges and no loops. Ok

3-The user chooses **4** as the root. Convert the tree to a rooted tree with **4** is the root



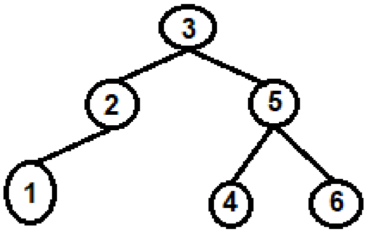
4-Convert the tree to a binary tree



5-Traverse the binary tree inorder: 1,2,3,6,5,4

6-Insert the traversed nodes one-by-one into a balanced binary search AVL tree.

Final output



7-Traverse the final tree inorder, postorder and preorder.

Inorder: 1,2,3,4,5,6

Postorder: 1,2,4,6,5,3

Preorder: 3,2,1,5,4,6