|  |  |
| --- | --- |
| **Data Structures & Algorithms**  Diploma in CSF, IT  Year 2/3 (2020/21) Semester 4/6 | **Week 12** |
| **2 Hours** |
| **Tutorial 10 – AVL Trees** | |

1. An AVL tree is to be created with the items given below.

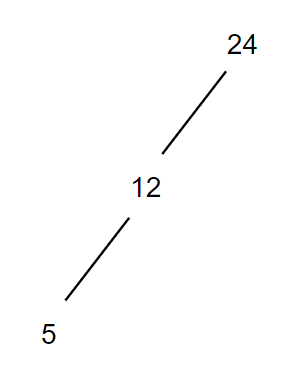
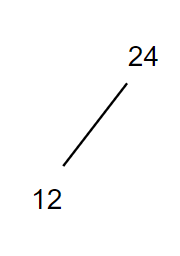
24, 12, 5, 30, 20, 45, 11, 13, 9, 16

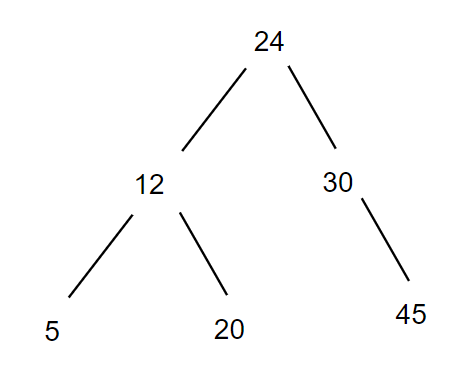
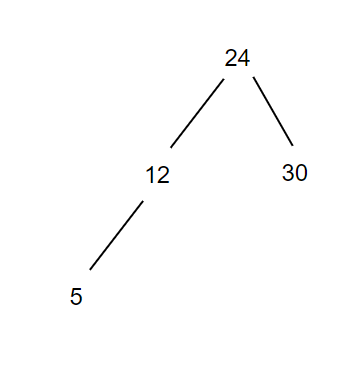
Assuming the items are inserted in the order given (i.e. from left to right),

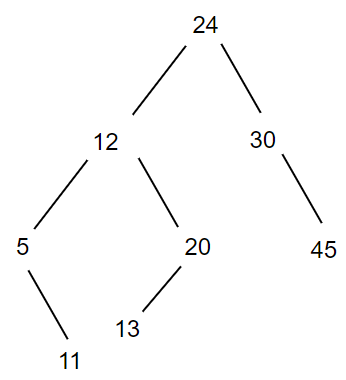
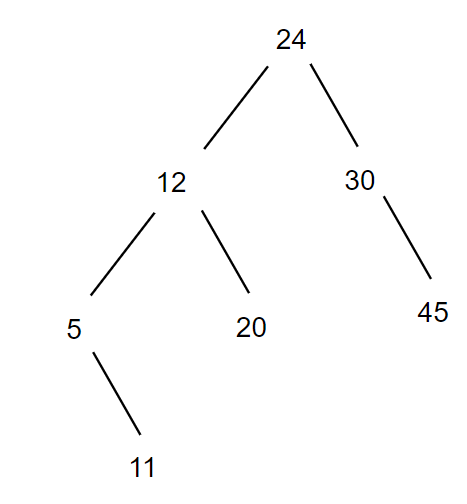
1. Show the intermediate trees at each stage of rotation.

Explain your workings.







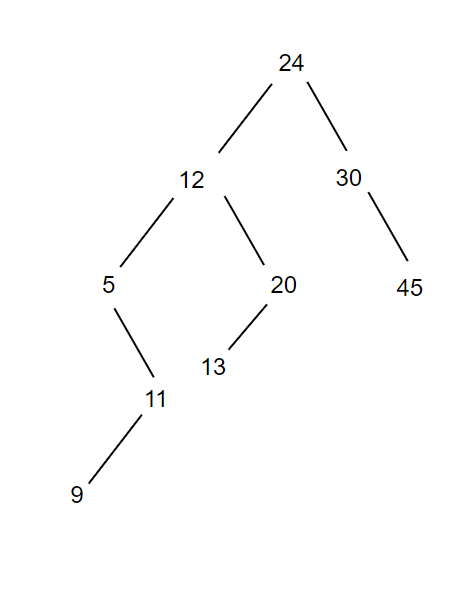


Tree unbalance at 5

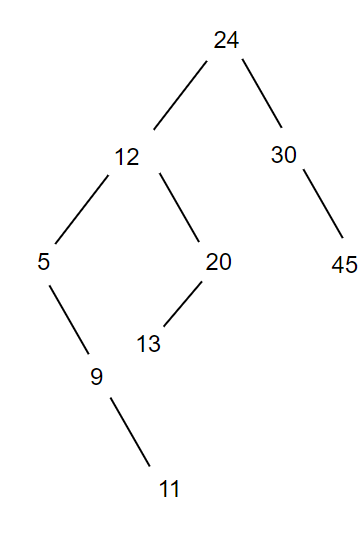
Subtree of 5 is right heavy and

Its right subtree is left heavy

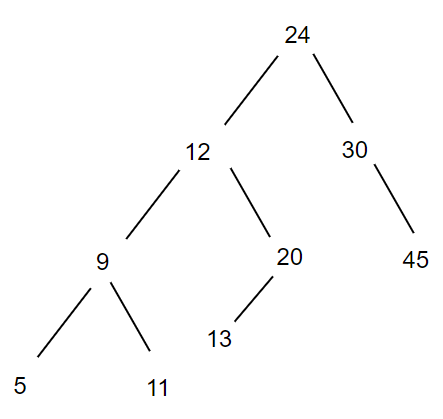
Right-left rotation



Right rotation at Node 11



Left rotation at node 5

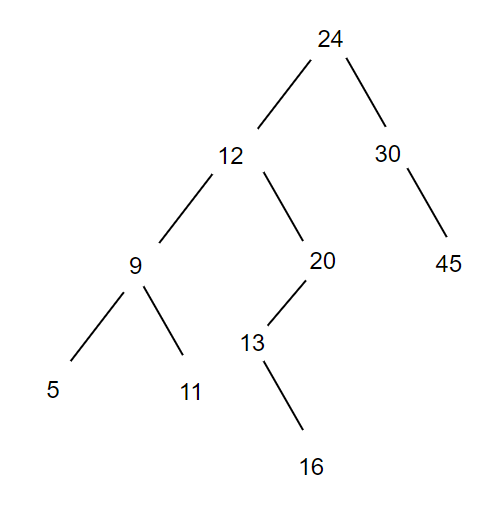


Tree is unbalance at node 20

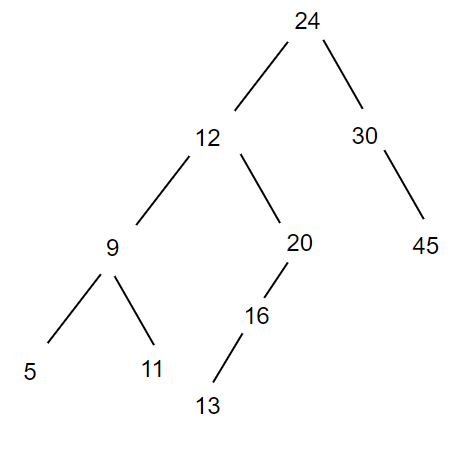
Node 20 is left heavy

And its left heavy node 13 is right heavy

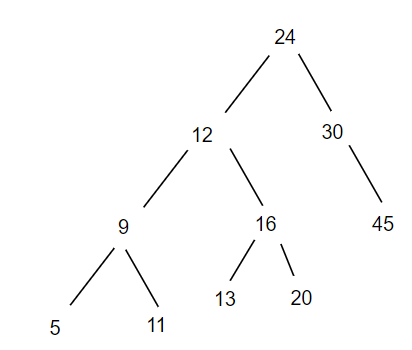
Left-right rotation



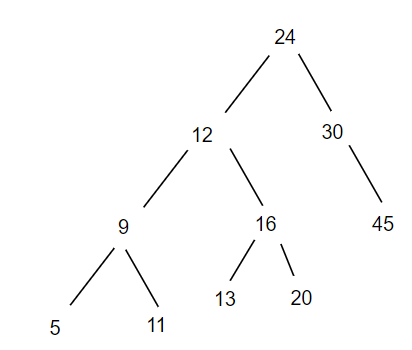
Node 13 left rotation



Node 20 right rotation



1. Draw the final AVL tree.



1. Write a method that will traverse the final tree in the following manner:

5 11 9 13 20 16 12 45 30 24

You may assume that the node of the tree has been defined as follows:

struct AVLNode

{

ItemType item;

AVLNode\* left; // left subtree

AVLNode\* right; // right subtree

}

Void Postorder(BinaryNode\* t){

If (t is not empty){

Traverse(left subtree of t)

Traverse(right subtree of t)

Process node

}

}

1. An AVL tree is to be created with the items given below. Draw the final AVL tree.

|  |  |
| --- | --- |
| INSERT | 7 |
| INSERT | 2 |
| INSERT | 8 |
| INSERT | 6 |
| INSERT | 9 |
| **INSERT** | **3** |
| **INSERT** | **4** |
| **INSERT** | **5** |
| **DELETE** | **9** |

