Assignment - 04

NAME :- CH. Nagi Reddy

REG. NO :- 192324170

COURSE :- Data Structure

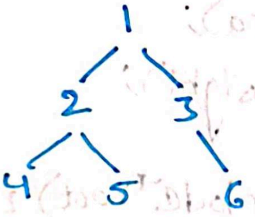
COURSE CODE :- CSA0389

1. Develop a C program to implement the tree traversals (Inorder, Preorder, Postorder)

```
#include <stdio.h>
#include <stdLib.h>
Struct Node {
    int data;
     Struct Node * left;
     Struct Node* right;
 };
Struct Node* CreateNode (int data) {
      Struct Node* newNode = (struct Node*)malloc (sizeof(struct
                                                          Node));
      newNode → data = data;
      newNode → left = Null;
      newNode → right = NULL;
      return newNode'
 }
 void inordertraversal (struct Node* root) {
      if (root == NuLL)
           return;
      inordertraversal (root → left);
       Printf ("%d", root → data);
      inordertraversal (root → right);
 }
 void Preordertraversal (struct Node*root) {
   if (root == NULL)
         return;
    Printf (" %d", root → data);
    Preorder-traversal (root → left);
     Preorder traversal ( root → right);
      Print f (" % d", root → data);
   }
```

```
int main() {
    Struct Node * root = CreateNode (1);
    root → left = CreateNode (2);
    root → right = CreateNode (3);
    root → left → left = CreateNode (4);
    root → left → right = CreateNode (5);
    root → right → right = CreateNode (6);
    Printf ("Inorder Traversal:");
    inorderTraversal (root);
    Printf ("\n");
    Printf ("Preorder Traversal:");
    PreorderTraversal (root);
    Printf ("\n");
    Printf (" Postorder Traversal:");
    PostorderTraversal (root);
    Printf ("\n");
    return 0;
}
```

Input: Creating the tree



Output:

```
Inorder Traversal: 4 2 5 1 3 6
 Preorder Traversal: 1 2 3 4 5 6
 Postorder Traversal: 4 5 2 6 3 1
```

2. Construct AVL Tree for the following elements 3,2,1,4,5,6,7 followed by 10 to 16 in reverse Order.

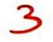
Sol:- To Construct an AVL Tree for the given elements.

Elements to insert

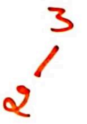
- First Sequence : 3, 2, 1, 4, 5, 6, 7
- Second Sequence (reverse Order) ! 16, 15, 14, 13, 12, 11, 10

Steps to Construct AVL Tree :

1. Insert 3 !

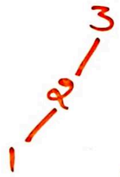


2. Insert 2 :



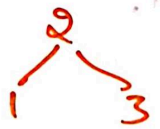
* Balance factor for node 3 is 1 , so no rotation needed.

3. Insert 1 !

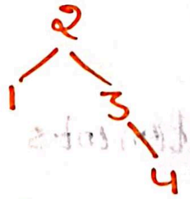


* Balance factor for node 3 is 2, and node 2 is 1, so we need a right rotation at node 3.

After rotation, the tree becomes:

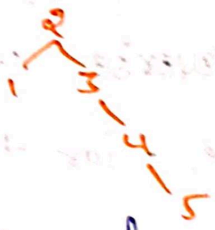


4. Insert 4 :



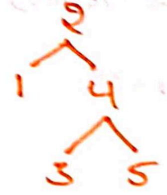
* Balance factor for node 2 is 0, so no rotation needed

5. Insert 5

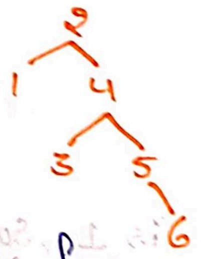


-* Balancing factor for node-2 is andso node 4 is -1, so we need a left rotatation at node 3

After rotation!

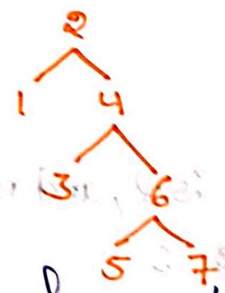


6. Insert 6:



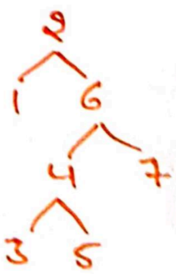
*Balance factor for node 4 is -1, so no rotation needed

7. Insert 7:



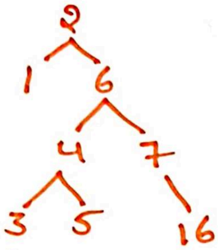
* Balance factor for node 4 is -2 and node 6 is -1, So we need left rotation at node 4.

After rotation:



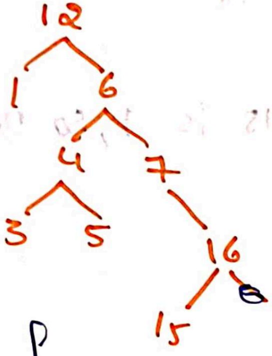
Next, we will insert the elements 16, 15, 14, 13, 12, 11, 10 in reverse order.

8. Insert 16:



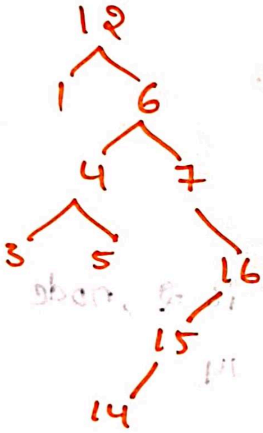
*Balance factor for node 7 is -1, so no rotation needed

9. Insert 15:



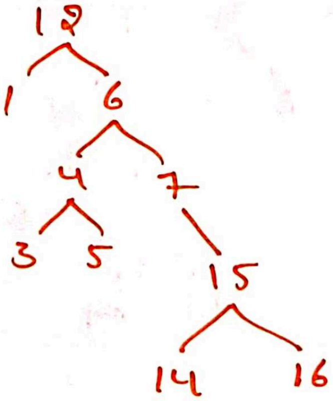
* Balance Factor for node 16 is 1, so no rotation needed

10. Insert 14:

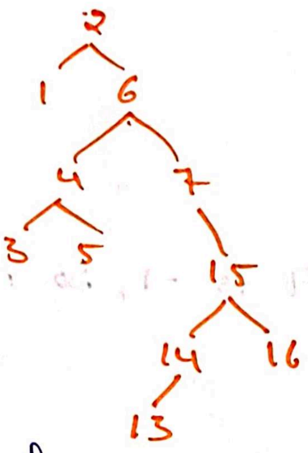


*Balance factor for node 16 is 2, node 15 is 1, so we need a right rotation at node 15.

After rotation:

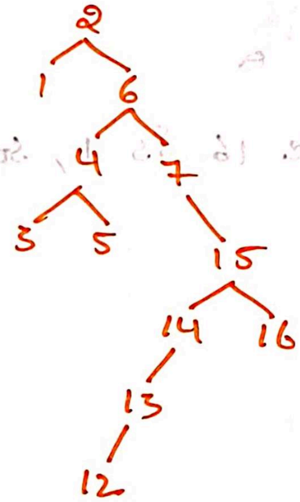


11. Insert 13:



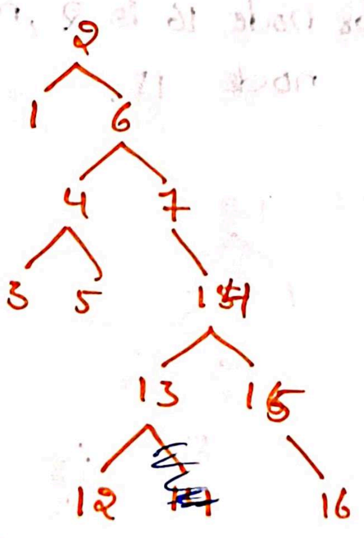
*Balanc factor for node 15 is 1, so no rotation needed

12. Insert 12:

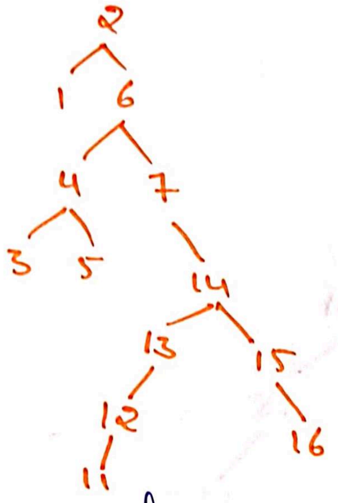


*Balance factor for node 15 is 2, node 14 is 1, so we need a right rotation at node 14

After rotation

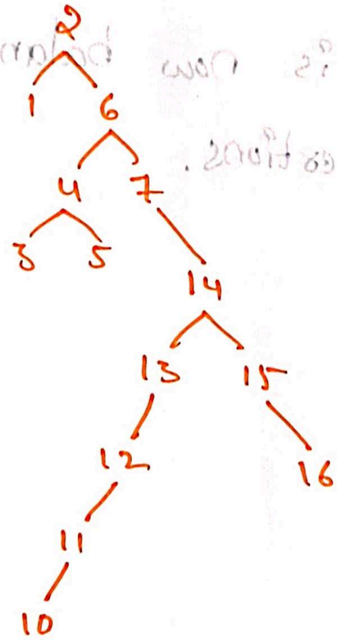


13. Insert 11:



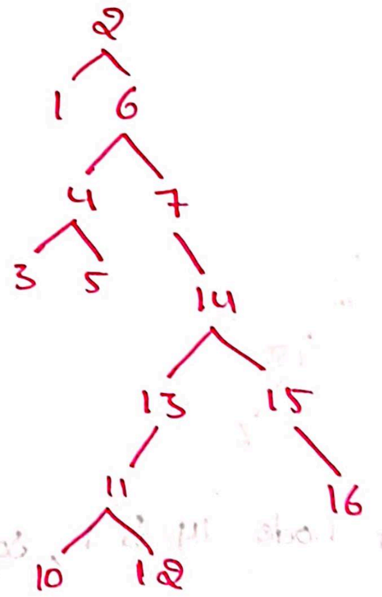
* Balance factor for node 14 is 1, so no rotation needed.

14. Insert 10:



* Balance factor for node 14 is 2, node 13 is 1, so we need a right rotation at node 11.

After rotation, The final Tree :



This AVL Tree is now balanced with given sequence of insertions.