Assignment - 5

Name :- G. Harshitha

Reg No :- 192324250

Department :- CSE(AI&DS)

Course :- Data structure.

Course Code :- CSA0389

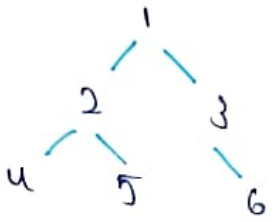
Develop a C Program to implement the tree Traversals (Inorder, Preorder, postorder).

```
#include <stdio.h>
#include <stdlib.h>
  Structure Node {
      int data;
    Struct Node * left;
    struct node * right;
};
struct node * create node (int data){
  struct node* newnode = (struct node*). malloc
                    (size of (struct node));
  newnode -> data = data;
  new node -> left = NULL;
  newnode -> right = NULL;
  return newnode;
}
 void inordTraversal (struct node * root) {
     if (root == NULL)
        return;
   inorder Traversal (root -> left);
   print f ("%d", root -> data);
   inorder Traversal (root -> right);
 }
 void preorder Traversal (struct node * root){
     if (root == NULL)
```

```
    return;
    printf("%d", root->data);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}
void postorderTraversal(struct node* root){
    if (root == NULL)
        return;
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%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 4 5 3 6

postorder Traversal : 4 5 2 6 3 1.

5)

construct AVL tree for the following elements 3, 2, 1, 4, 5, 6, 7 followed by 16 to 11 in reverse order.

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 the AVL Tree:

1. Insert 3 :-

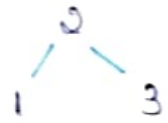
2. Insert 2 :-

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

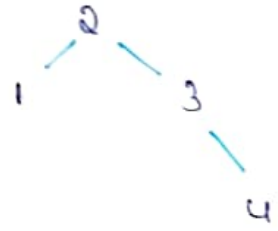
3. Insert

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

* After rotation, the tree becomes

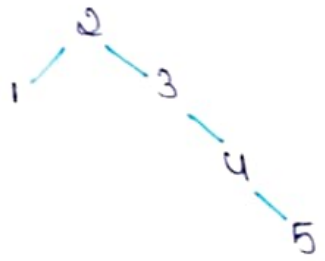


4. Insert 4:



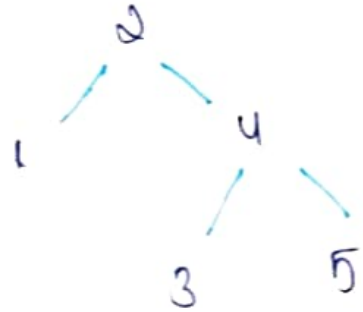
* Balance factor for node 2 is 0, so no rotation needed.

5. Insert 5:-

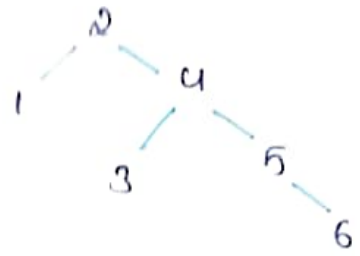


* Balancing factor for node 3 is -2, and node 4 is -1, so we need a left rotation at node 3.

* After rotation:

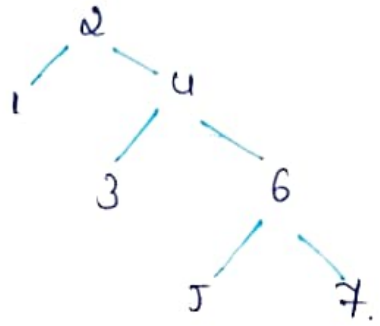


Insert 6:



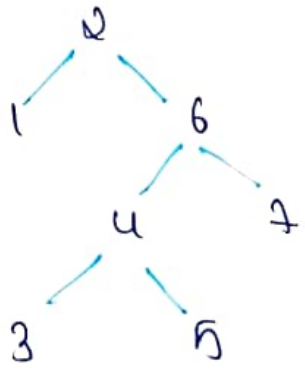
* Balancing factor for node u is -1, so no rotation needed.

Insert 7:



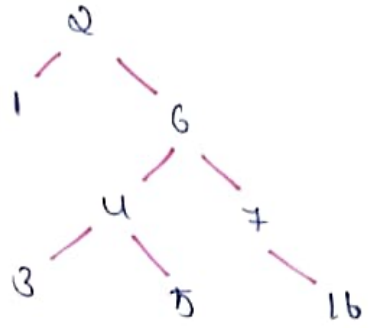
* Balancing factor for node u is -2 and node 6 is -1, so we need left rotation at node u.

After rotation:



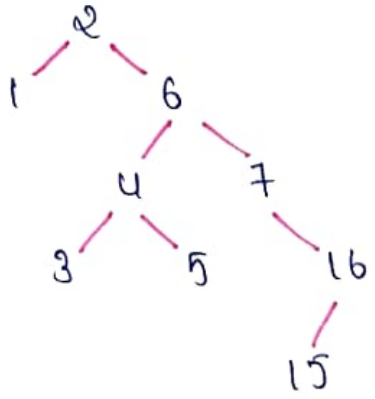
Next, we will insert the element 16, 15, 14, 13, 12, 11, 10. in reverse order.

Insert 16



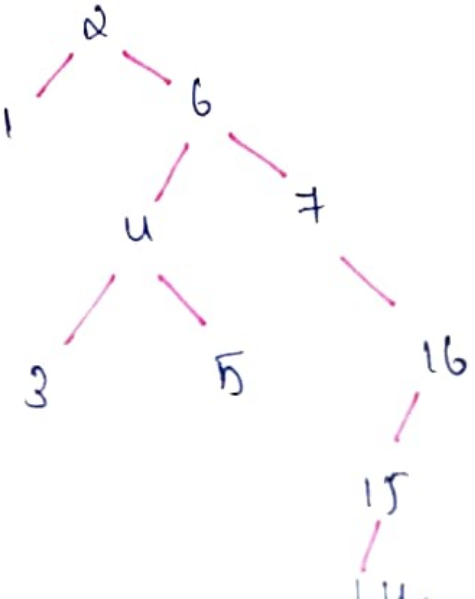
* Balance factor for node 7 is -1, so no rotation needed.

9. Insert 15:



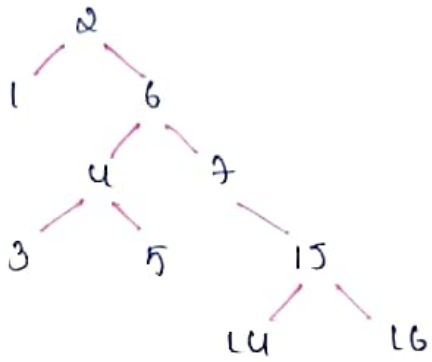
* Balance factor for node 16 is -1, so rotation needed.

10. Insert 14.

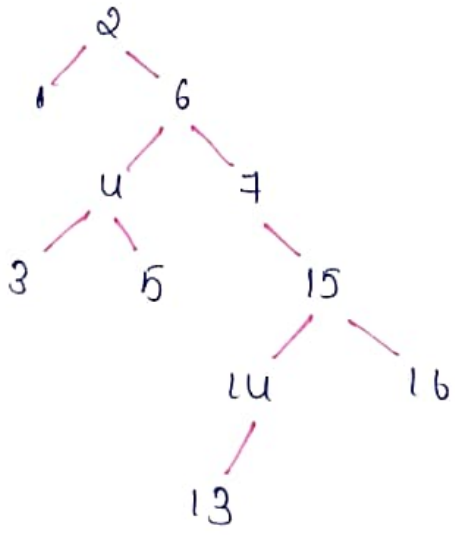


* Balance factor for node 16 is 2, node 15 is 1, so we need a right rotation at node 15.

After rotation:

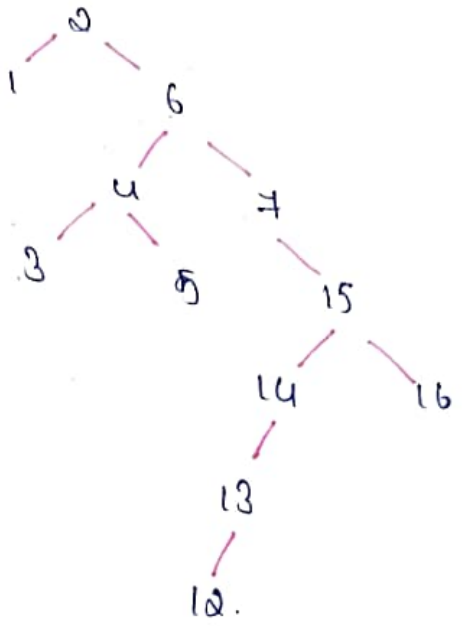


11. Insert 13:



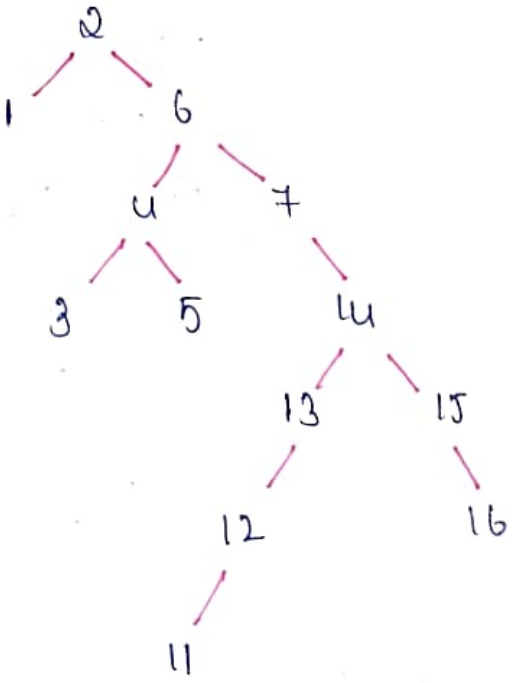
* Balance factor for node 15 is 1. So, no rotation needed.

12. Insert 12



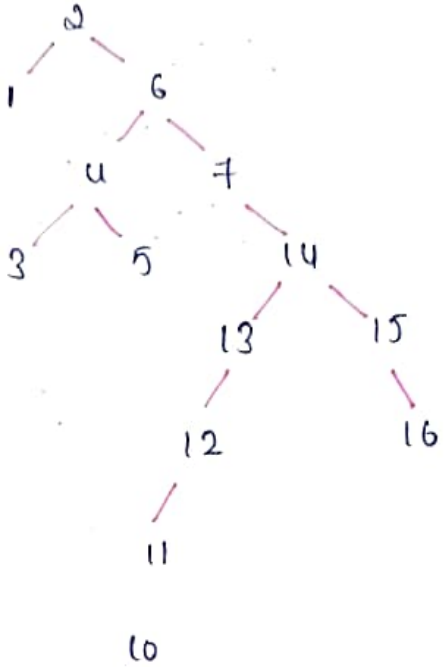
* Balance factor for node 15 is 2, node 14 is 1, so we need a right rotation at node 14.

13. Insert 11.



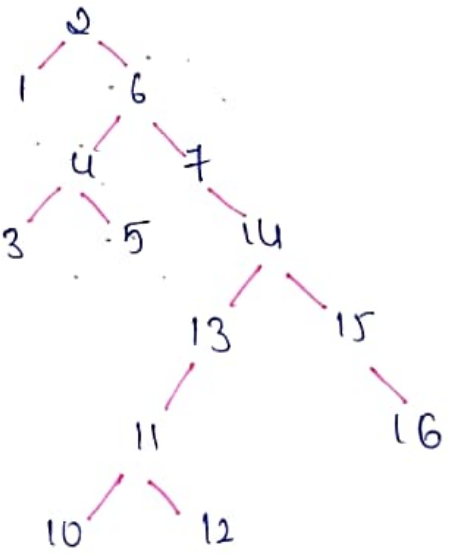
* Balance factor for node 14 is 1, so no rotation needed.

14. Insert 10:



* Balance factor for node 4 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.