ASSIGNMENT - 04

NAME: M. Prasad

REG NO: 192311252

COURSE CODE: CSA 0389

COURSE NAME: DATA STRUCTURE

DATE: 21-Aug-2024

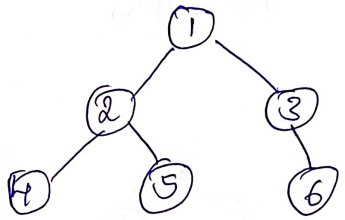
op a cprogram to implement the tree
versal (inorder, preorder, postorder):

```
#include <stdio.h>
#include <stdlib.h>
 struct node {
        int data;
        struct node* left;
        struct node* right;
    };
    struct node* create node (int data) {
    struct node* newnode = (struct node*)
malloc (sizeof (struct node));
        New node -> data = data;
        new node -> left = NULL;
        new node -> right = NULL;
        return newnode;
    }
    void inorder traversal (struct Node* root) {
        if (root == NULL)
            return;
        inorder traversal (root -> left);
            printf ("%d root -> data);
        inorder traversal (root -> right);
    }
```

```
void preorder traversal (struct Node* root){
    if (root==NULL)
        return;
    Print ("%d", root->data);
    preorder Traversal (root->left);
    preorder Traversal (root->right);
}
void postorder traversal (struct Node* root){
    if (root==NULL)
        return;
    post order traversal (root->left);
    post order traversal (root->right);
    print ("%d", root->data);
}
int main(){
    struct Node* root=(create Node(1);
        root->left =create Node(2);
        root->right = create Node(3);
    root->left->left =create Node(4);
    root->right->left =create node(5);
    root->right->right = create Node(6);
printf("Inorder Traversal");
    inorder Traversal (root);
    print ("/n");
printf("preorder traversal:");
    preorder (Traversal (root);
    print ("/n");
```

```
printf("postorder Traversal:");
postorder Traversal(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

construct AVL tree for the following elements 3, 2, 1, 4, 5, 6, 7 followed by 10 to 16 in reverse order.

To construct an AVL tree for the given elements.

Elements to Insert:-

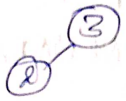
* first sequence 3, 2, 1, 4, 5, 6, 7
* second sequence (reverse order) 16, 15, 14, 13, 12, 11, 10

steeps to construct the AVL Tree:

① Insert 3:

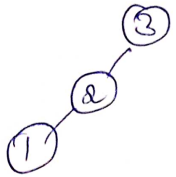
3

2. Insert 2:

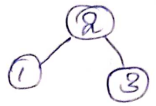


· Balance factor for node 3 is 1 so, on rotation needed

3. Insert 1

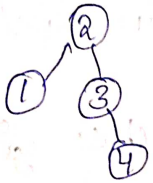


* Balance factor for node 3 is 2, and node 2 is 1, so we need a right rotation at node 3.



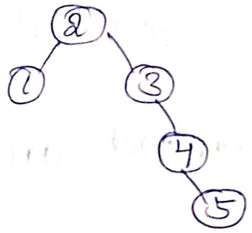
* After rotation

4. Insert 4:

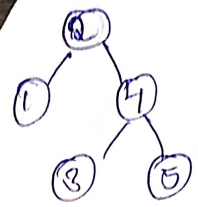


Balance factor for node 2 is 0 so, no rotation needed

5. Insert 5:-

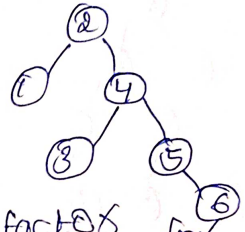


* Balance factor for node 3 is -2, node 4 is -1, so we need LR at 3



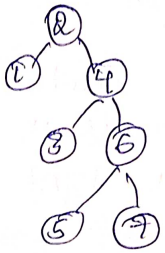
After rotation

6. Insert 6:

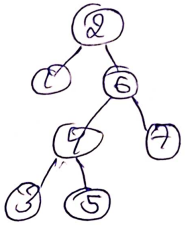


*Balance factor for node 4 is -1, so no rotation needed.

7. Insert 7:-



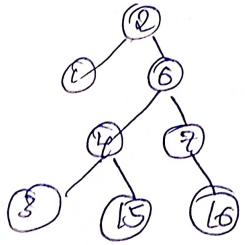
* Balance factor for node 4 is -2 & node 6 is -1 and we need to LR is need.



* After rotation.

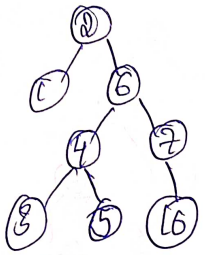
Next, we will insert the elements 16, 15, 14, 13, 12, 11, 10, in reverse order.

⑧ Insert 16.



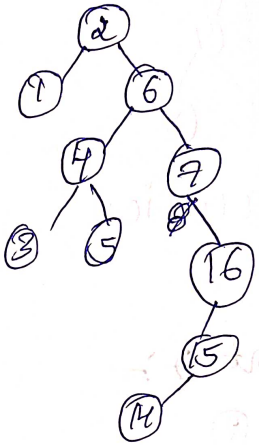
*Balance factor for node 7 is -1, so no rotation needed.

⑨ Insert 15:-



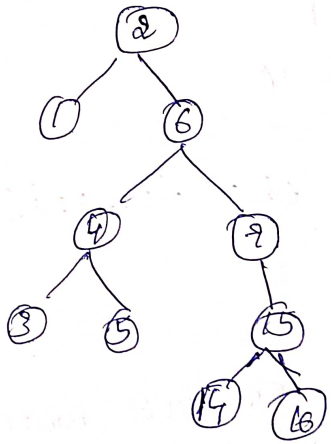
* Balance factor for node 16 is 1, so no rotation is needed

10. Insert 14

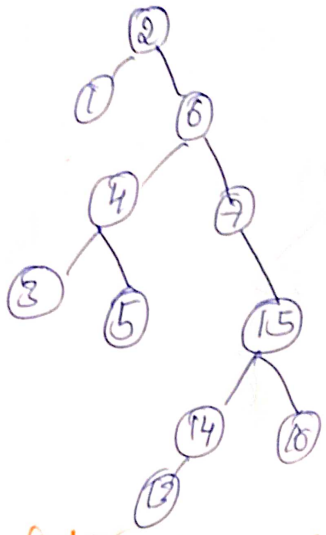


* Balance factor for node 16 is 2 node 15 is 1, so we need a right rotation at node 15.

After rotation:-

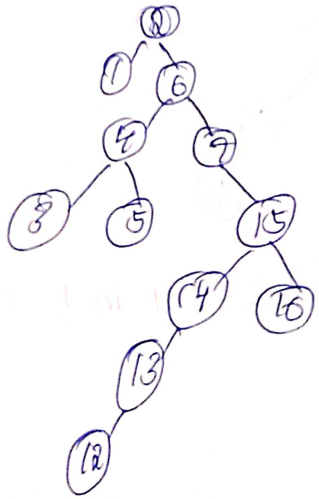


sert 13:

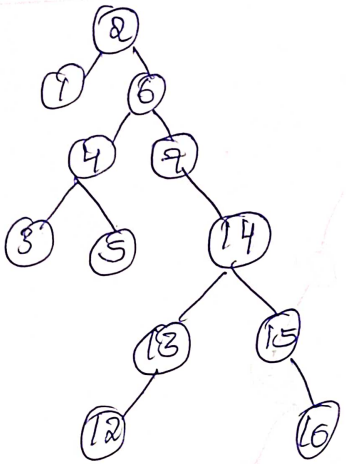


* Balance factor for node 15 is 1, so, no rotation needed

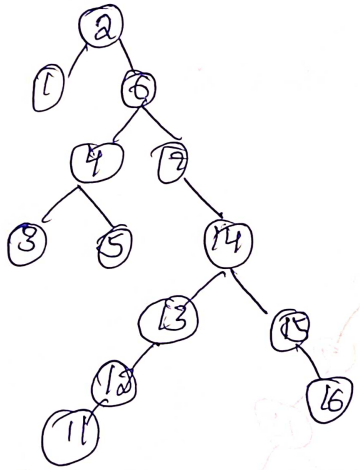
12. Insert 12



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

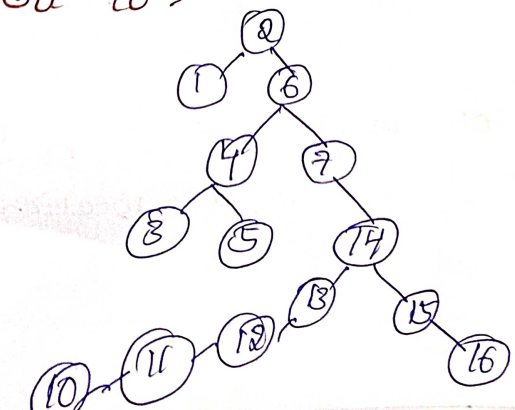


13. Insert 11 :-



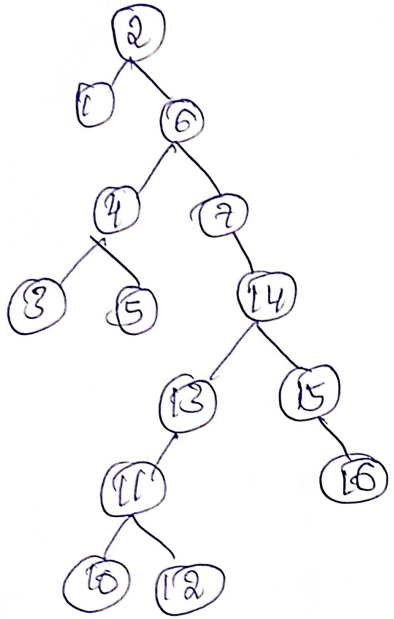
* Balance factor for node 14 is 1, so no rotation needed

14. Insert 10 :-



ance factor for node 14 is 2, node 13 is 1, we need to RL at node 11.

After rotation, final tree :-



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