Assignment - 04

Name :- T. Sanhith

Regno :- 192311228

Course code :- CS40389

Course name :- Data structure

Date :- 21-Aug-2024

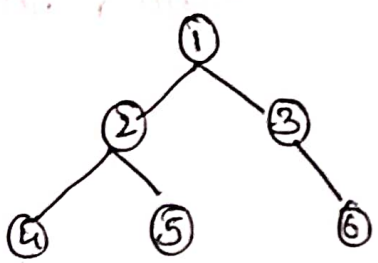
Develop a c program to implement the tree Traversal (inorder, Preorder, postorder):

```
#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(size of (struct node));
    Newnode → data = data;
    newnode → left = Null;
    newnode → right = Null;
    return newnode;
}
void inorder Traversal (struct Node * root){
    if (root == NULL)
        return;
    inorder traversal( root → left);
    Printf("%d root→ data)",
    in order traversal (root →right);
}
```

```
void Preorder traversal (struct Node *root){
    if (root == NULL)
        return;
    printf("%d", root -> data);
    Preorder Traversal (root -> left);
    preorder traversal (root -> right);
}
void postorder traversal (struct Node * root){
    if (root == NULL)
        return;
    postorder traversal (root -> left);
    postorder traversal (root -> right);
    Print("%d", root -> data);
}
int main(){
    struct Node* root = (reate Node(1);
        root -> left = (reate Node(2);
        root -> right = (reate Node (3);
        root -> left -> left = (reatenode (4);
        root -> right -> left = (reate no !e (5);
        root -> right - right = (reatenode(6);
    printf(" Inorder Traversal");
    inorder Traversal (root);
    Print("\n");
    Printf(" preorder traversal :");
    preorder traversal (root);
    Printf("\n");
```

```
Print f(" Postorder Traversal : ");
Postorder Traversal (root);
    Print f("\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

② 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

steps to construct the AVL Tree :-

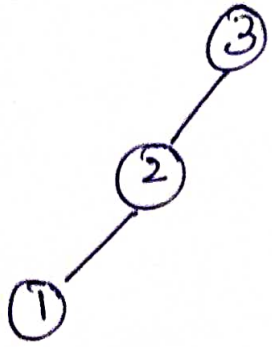
① Insert 3 :

3

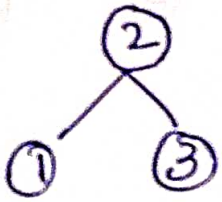
② insert 2 :

• Balance factor for node 3 is
so, no rotation needed

③ Insert 1

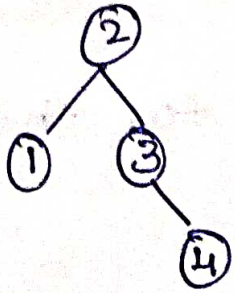


* Balance factor for node 3 is 2, and node 2 is 1, so we need a right rotation at node 3.



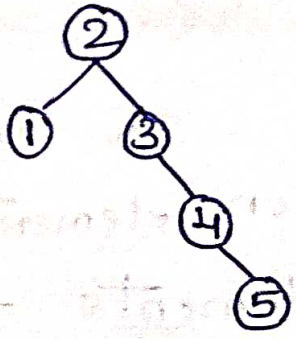
* After rotation.

④ Insert 4 :

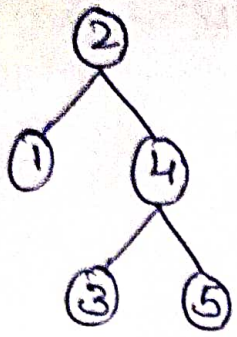


Balance factor for node 2 is 0
so, no rotation needed

⑤ insert 5 :-

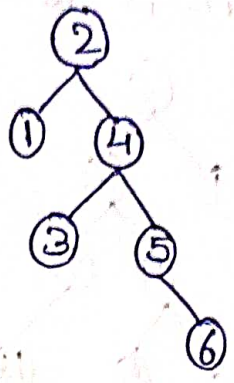


* Balance factor for node 3 is -2, node 4 is -1, so we need a LR at 3



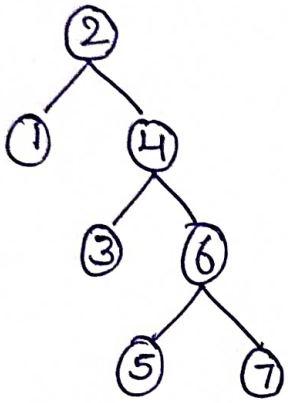
* After rotation

⑥ Insert 6 :

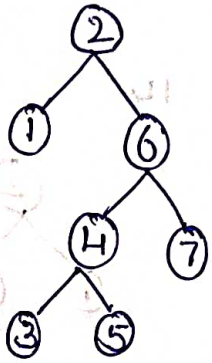


* Balance factor for node 4 is -1, so no rotation needed

⑦ Insert 7 :-



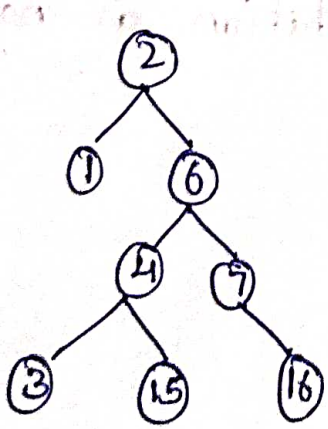
* Balance factor for node 4 is -2 & node 6 is -1 and we need to LR is need



* After rotation.

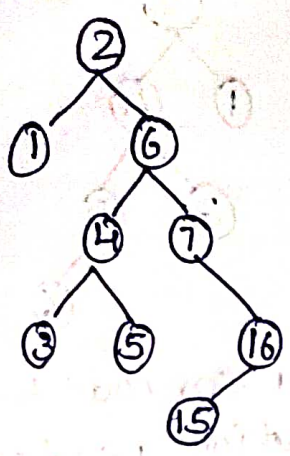
Next, we will insert the elements 16, 15, 14, 13, 12, 11, 10 in reverse order

⑧ Insert 16.



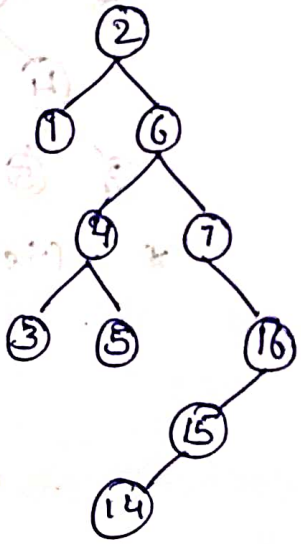
* Balance factor for node 7 is -1, so, no rotation needed

⑨ Insert 15 :-



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

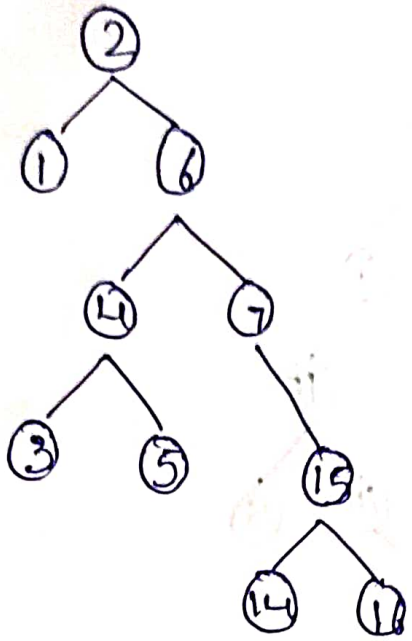
⑩ Insert 14



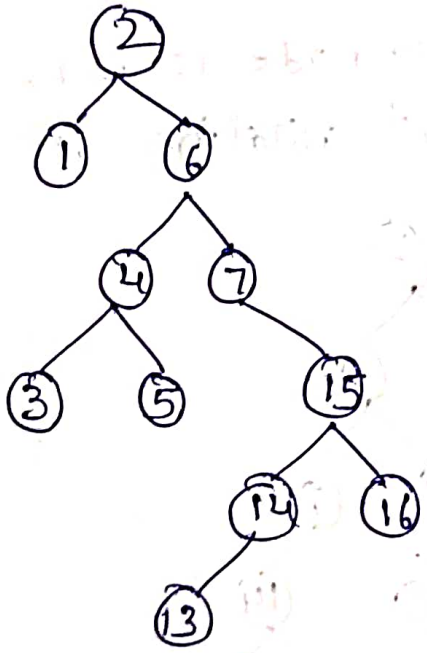
* Balance factor for node 16 is 2, node 15 is 1 so, we need a right rotation at node 15

⑩ Insert 14

After rotation:-

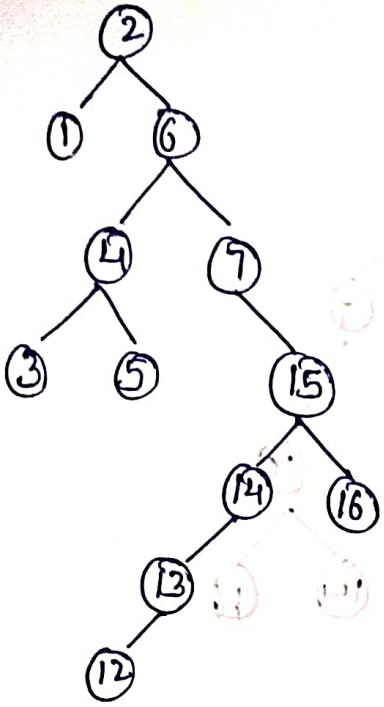


(18) Insert 13 :

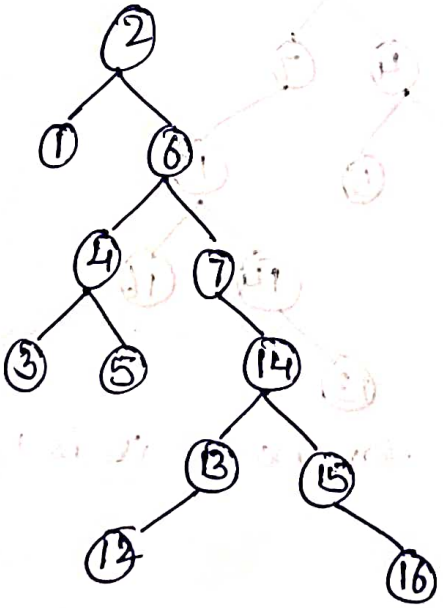


* Balance factor for node 15 is 1, so, no rotation needed

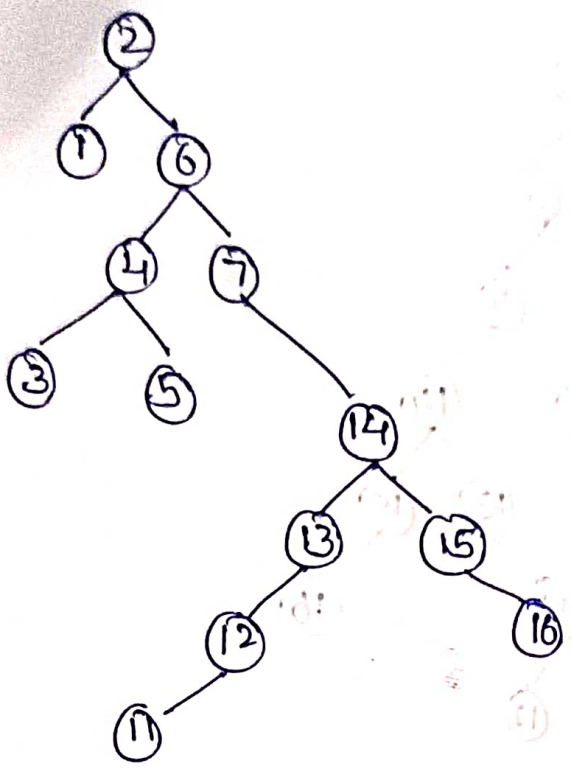
(12) Insert 12



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

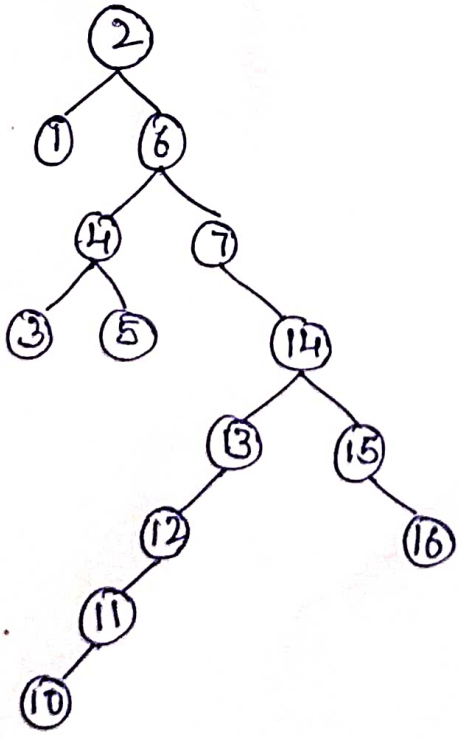


(13) Insert 11 :-



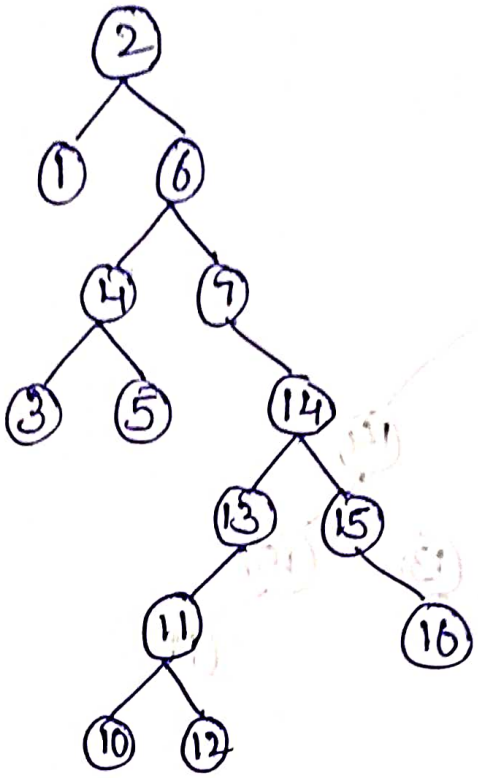
* Balance factor for node 14 is 1, so no rotation needed

(14) Insert 10 :



* Balance factor for node 14 is 2, node's is 1, so we need to RL at node 11.

After rotation, the final tree:-



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