ASSIGNMENT-05

Name	:	Greeshma.V
Reg No	:	192365012
Dept	:	CSE (cyber Security)
Course code	:	CSA0381
Course Name	:	Data Structures
Faculty Name	:	Dr. Ashok kumar
Assignment No	:	05
Date of Submission	:	21-Aug-2024
No. of pages	:	12.

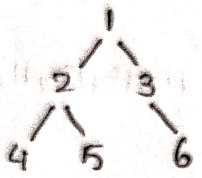
1. Develop a C program to implement the Tree Traversals (Inorder, preorder, postorder)

```
#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 inorder
    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);
    PreorderTraversal (root -> left);
    PreorderTraversal (root -> right);
}
Void postorderTraversal (struct Node* root){
    if (root == NULL)
        return;
    PostorderTraversal (root -> left);
    PostorderTraversal (root -> right);
    Print ("%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);
    Print("\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

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.

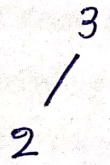
Elements to Insert

- First sequence: 3, 2, 1, 4, 5, 6, 7
- Second sequence (reverse order): 16, 15, 14, 13, 12, 11, 10.

Steps to Construct the AVL Tree:

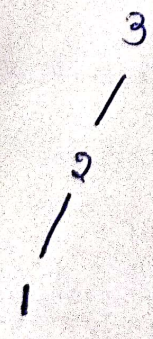
1. Insert 3:

2. Insert 2:



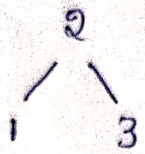
* Balance factor for node 3 is 1, so no rotation needed.

3. Insert 1

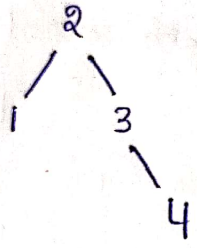


* Balance factor for node 3 is 2, and node 2 is 1, so we need a right rotation at node 3.

* After rotation, the tree becomes:

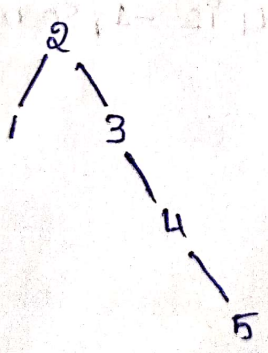


4. Insert 4:



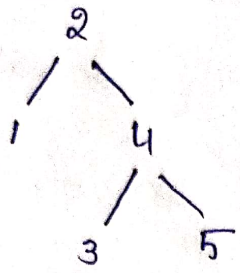
* Balance factor for node 2 is 0, so, no rotation needed.

5. Insert 5.

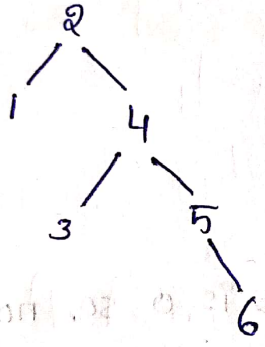


* Balancing factor for node 3 is -2, and node 4 is -1, so we need a left rotation at node 3.

* After rotation :

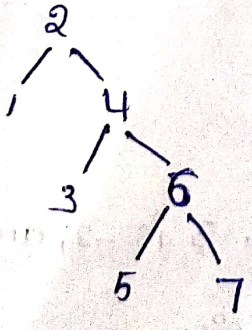


Insert 6 :



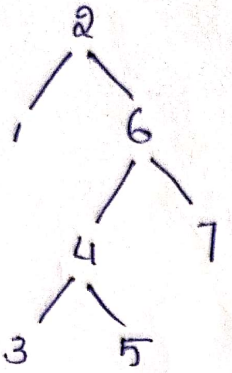
* Balance factor for node 4 is -1, so no rotation needed.

Insert 7 :



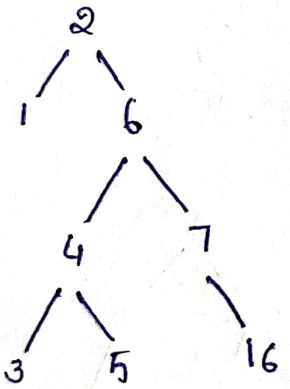
* Balance factor for node 4 is -2 and node 6 is -1, so we need left rotation at node 4.

After rotation:



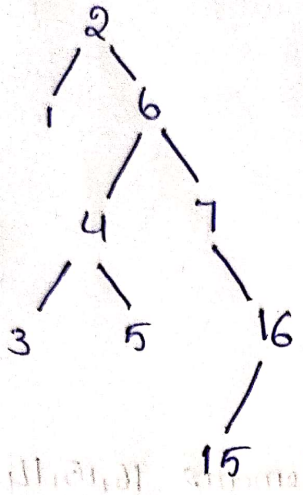
Next, we will insert the elements 16,15,14,13,12,11,10 in reverse order.

8. Insert 16.



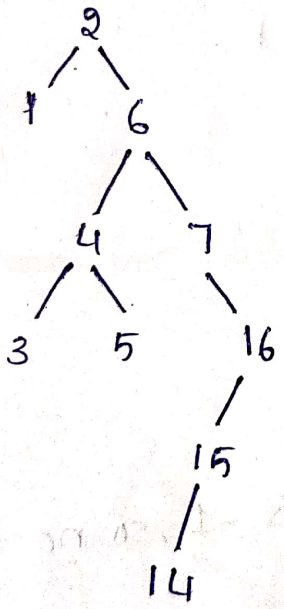
* Balance factor for node 7 is -1, so no rotation needed.

9. Insert 15:



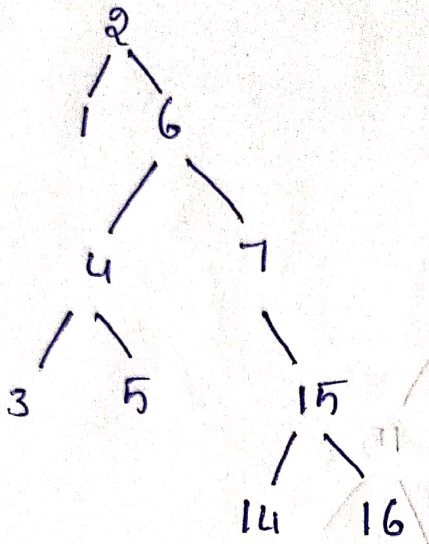
* Balance factor for node 16, is 1, so no rotation needed.

10. Insert 14.

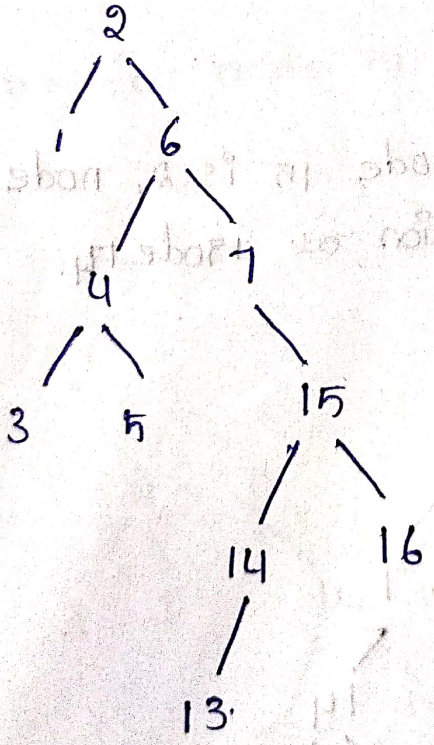


* Balance factor for node 16 is 2, node 15 is 1, so we need a right rotation at node 15.

After rotation.

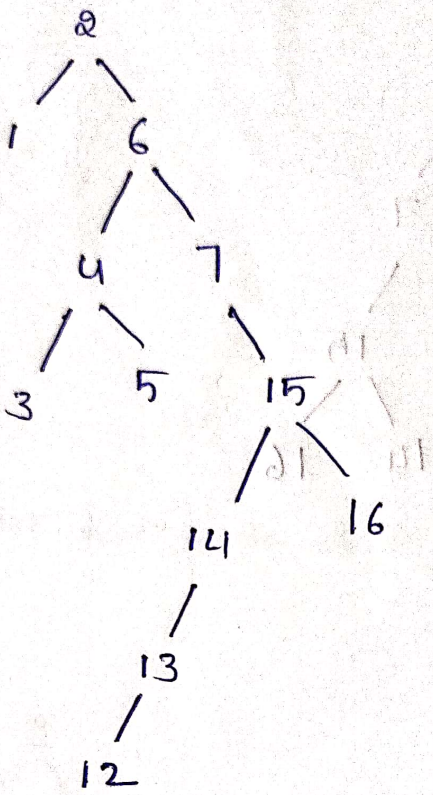


11. Insert 13:

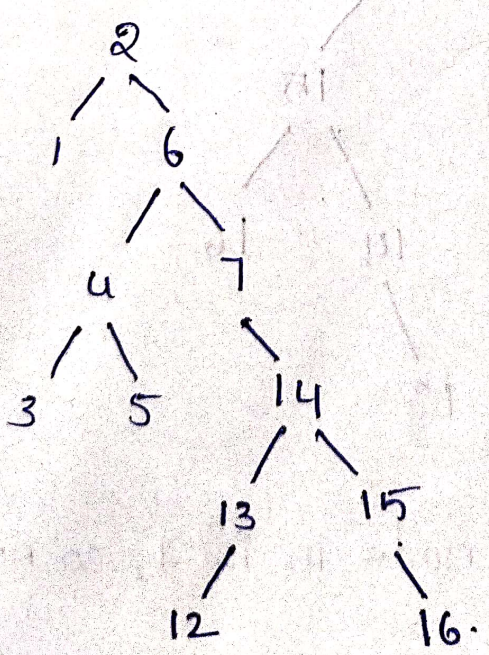


* Balance factor for node 15 is 1, so no rotation needed.

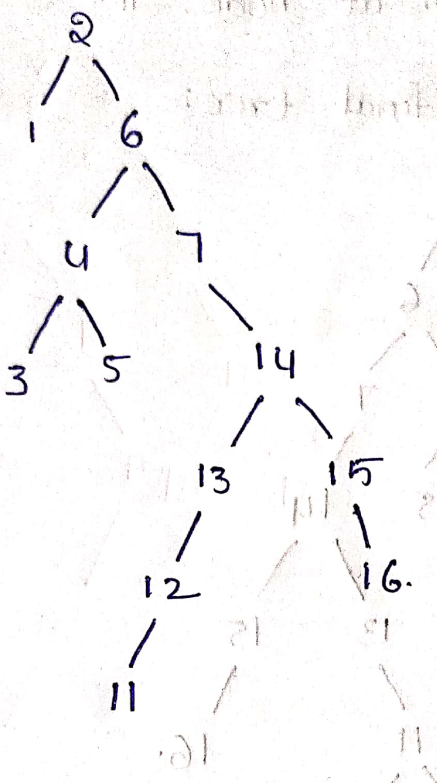
12, Insert 12



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

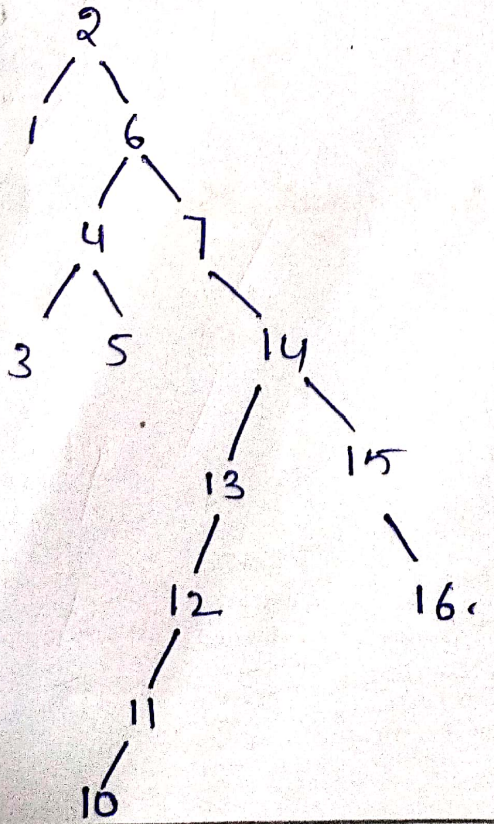


13. Insert 11.



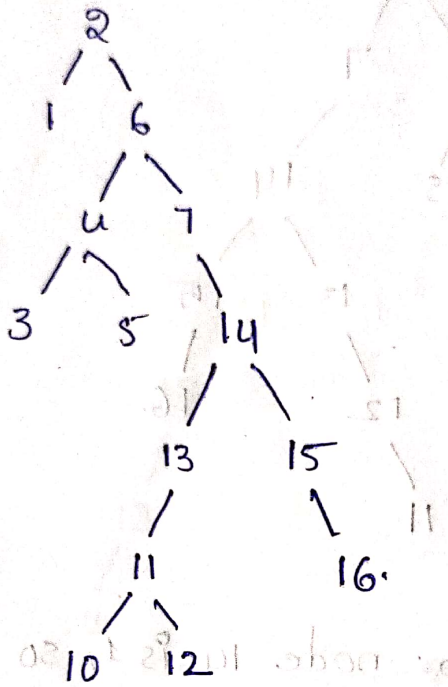
* Balance factor for node 14 is 1, so no rotation needed.

14. Insert 10;



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

After rotation, the final tree:-



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