Assignment - 04

Name	: M. Hiteshwar Reddy
Reg no	: 192325074
Dept	: AIML
Course code	: CSA0389
Course Name	: Data structure
Faculty Name	: Dr. Ashok Kumar
Date of Submission	: 21-08-24
No.of pages.	: 08

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

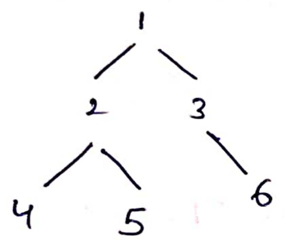
```
#include <stdio.h>
# include <stdlib.h>
    Struct  Node {
        int data;
        Struct Node* left;
        struct Node* right;
 };
Struct Node* create Node (int data) {
  Struct Node* newNode = (struct Node*) malloc (size of(struct
                                                     Node));
  newNode -> data = data;
  newNode -> left = NULL;
  newNode -> right = NULL;
    return newNode;
}
  void 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);

  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);
    printf("\n");
    printf(" preorder Traversal : ");
    preorder Traversal (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. Constuct 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.

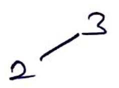
Elements to Insert

- 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:

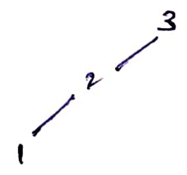
1. Insert 3 :

2. Insert 2:



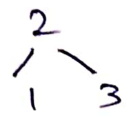
* Balance factor for node 3 is 1, so no rotation needed

3. Insert 1.

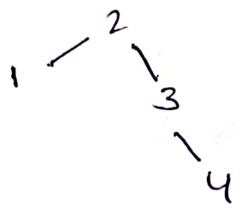


* Balance factor for node 3 is 2, & node 2 is 1, so we need a right rotation at node 3.

* After rotation, the tree becomes:

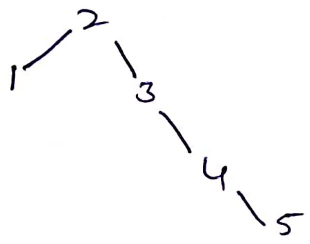


4. Insert 4!



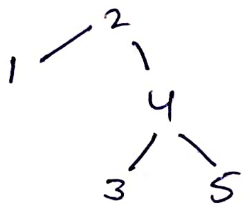
* Balance factor for node 2 is 0, so, no rotation needed

5. Insert 5.

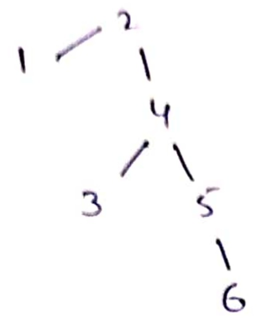


* Balancing factor for node 3 is -2, & node 4 is -1, so we need a left rotation at node 3.

After rotation:

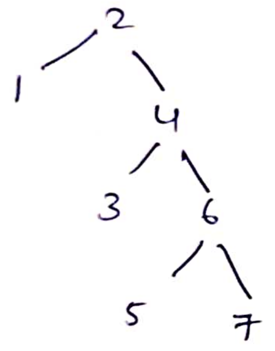


Insert 6:



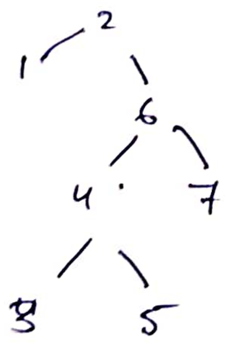
* Balance factor for node 4 is -1, so no rotation needed.

Insert 7:



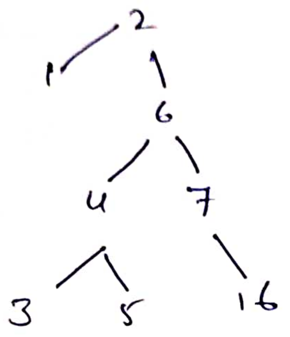
* Balance factor for node 4 is -2 & node 6 is -1, so we need left rotation at node 4.

After rotation:



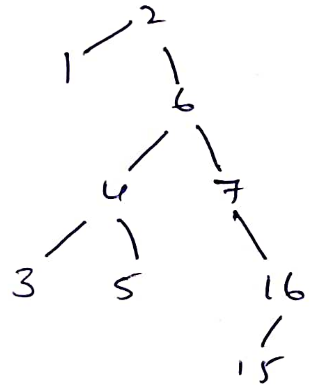
Next, we will insert the elements 16, 15, 14, 13, 12, 11, 10 in reverse order.

8. Insert 16



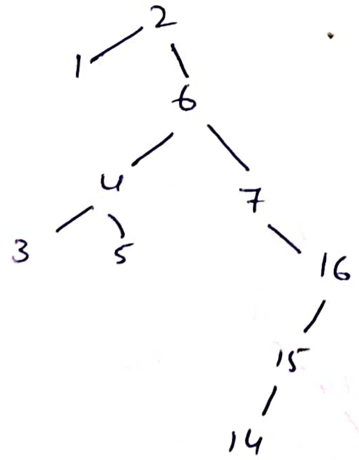
* Balance factor for node 7 is -1, so no rotation needed.

9. Insert 15:



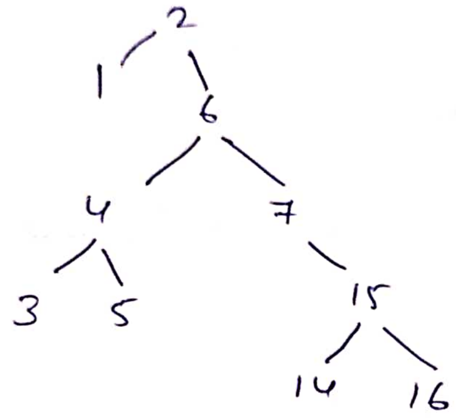
* Balance factor for node 16, is 1, so no rotation needed

10. Insert 14.

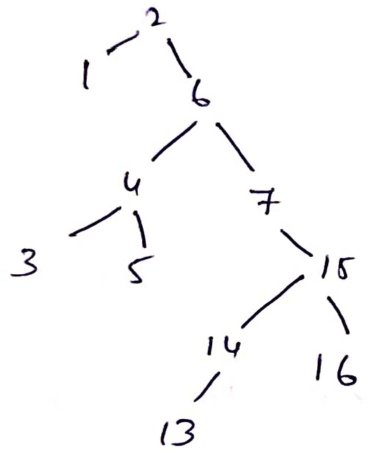


Balance factor for node 16 is 2, node 15 is 1, So we need a right rotation at node 15.

After rotation

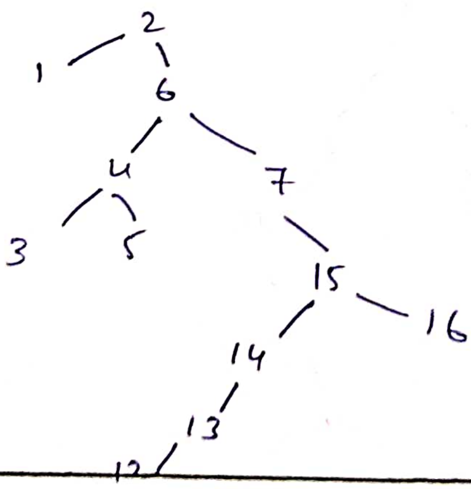


11) Insert 13:

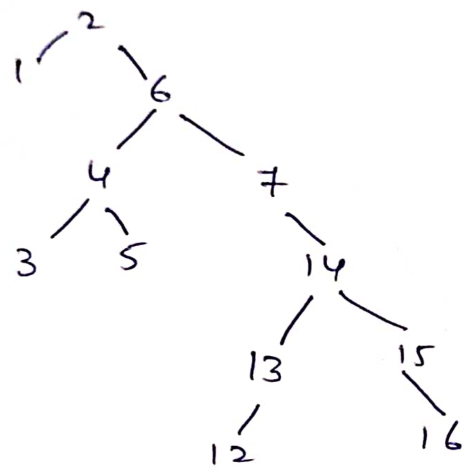


Balance factor for node 15 is 1, so no rotation needed.

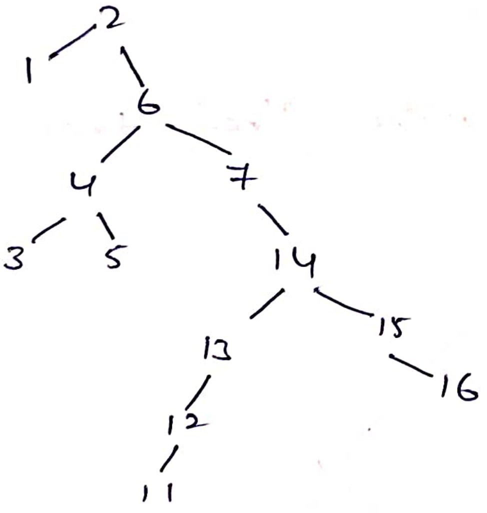
12) Insert 12



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

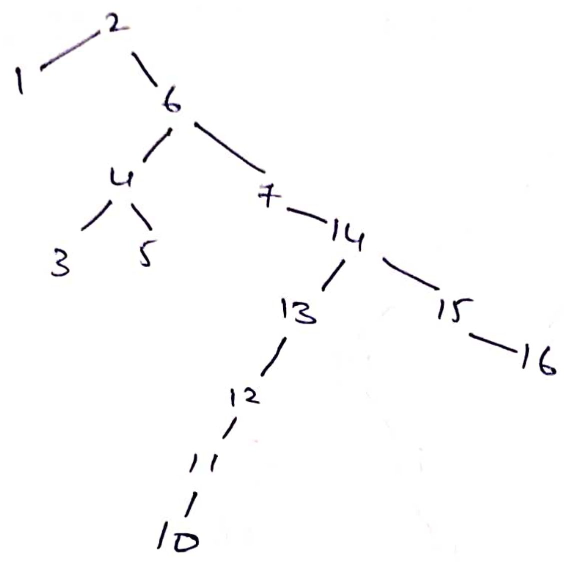


3) Insert 11



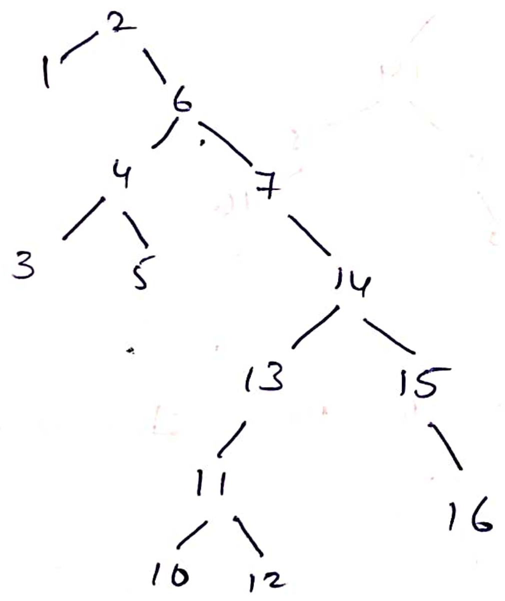
Balance factor node fo 14 is 1, so no rotation needed.

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 insertions.