ASSINGMENT-4

NAME :- M. Uday Kumar

DEPT :- AIML

SUB :- Data Structure

CODE :- CSA0389

REG :- 192325073

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;
 new node -> right = NULL;
 return newnode;
}
 void in order traversal (struct node * root) {
   if (root == NULL)
       return;
    in order traversal (root -> left);
    printf ("%d", root -> data);
  in order traversal (root -> right);
}
   void Pre order traversal (struct node * root) {
       if (root == NULL)
          return;
       Print f ("%d" root -> data);
       Pre order traversal (root -> left);
       Pre order traversal (root -> right);
}
```

```
void postorderTraversal (struct node * root ) {
    if (root == NULL)
        return;
    post order traversal (root -> left);
    post order traversal (root -> right);
    printf ("%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 -> left -> right = create node (5);
    root -> right -> right = create node (6);
    printf ("In ordered traversal:");
    in order traversal (root);
    printf ("\n");
    printf ("preorder traversal:");
    Pre order traversal (root);
    printf ("\n");
```

Out put:

In order traversal: 4 2 5 1 3 6

Pre order traversal: 1 2 4 5 3 6

Post order 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.

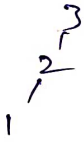
Step to construct the AVL tree:

1. Insert 3:

2. Insert 2.

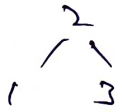
* Balance factor for node 3 is 1, so no rotation need.

3. Insert 1



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

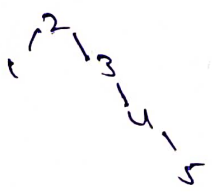
* After rotation



4. Insert 4

* Balance is 2 i.e 0, no rotation.

5. Insert 5



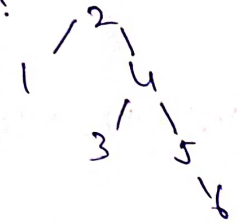
* Balancing factor for node 3 is 2

4 is -1

need rotation at node 3.

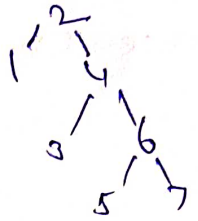
After rotation

Insert 6:

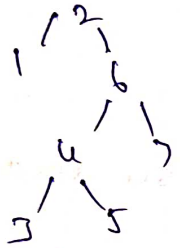


norotation

Insert 7 :

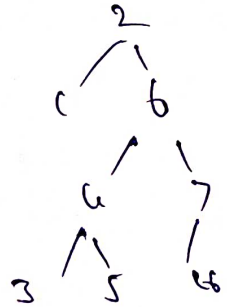


After rotation



Insert reverse order 16,15,14,13,12,11,10 in reverse.

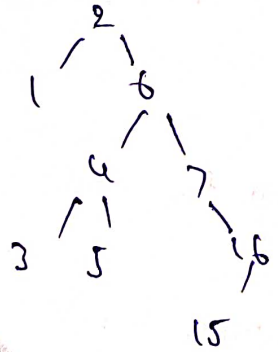
8. Insert 16



Bf = -1

NO rotation

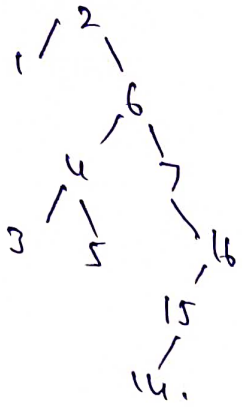
9. Insert 15:



BF = 1

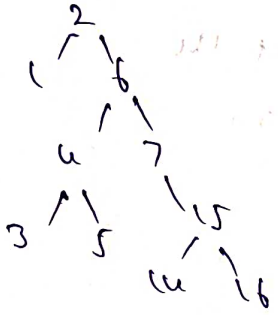
NO rotation.

Insert 14.

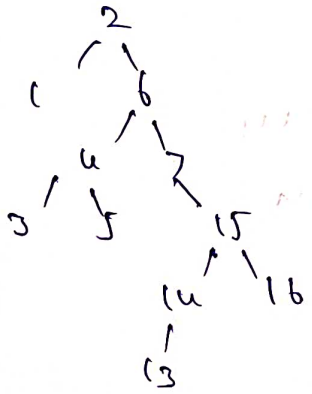


* Balance factor for node 16 is 2, node is 7 so we need a right rotation at node 15.

After rotation



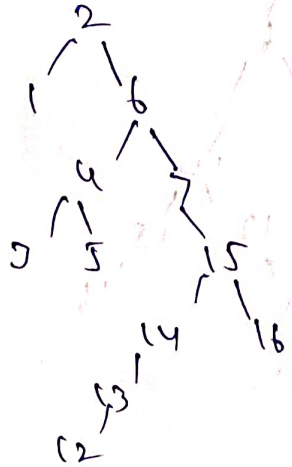
Insert 13



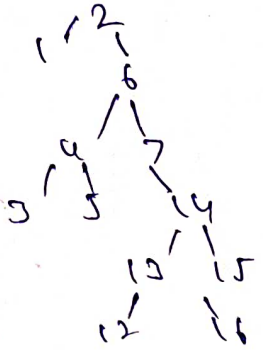
BF = 1

no rotation

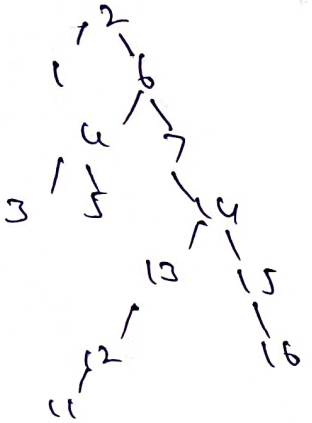
Insert 12



BF is 2 at 15.

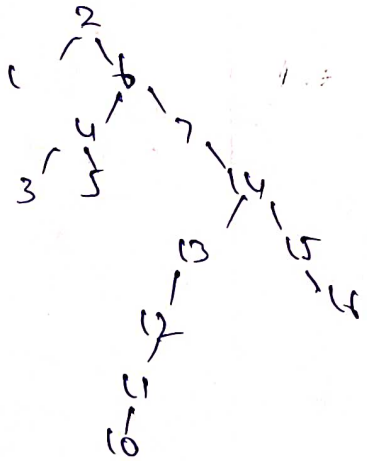


Insert 11.



BF = 1 at 14.
No rotation

Insert 10



BF = 2 at 14
Rotation

Final