**Ayush Goyal**

**190905522 CSE D 62**

**DAA Lab 7 (Week 7) – Transform and Conquer – I**

1. **Modify the solved exercise to find the balance factor for every node in the binary search tree.**

**CODE:**

#include<stdio.h>

#include<stdlib.h>

#define MAX(a,b) ((a)>(b) ? a : b)

typedef struct node{

    int val;

    struct node \*left;

    struct node \*right;

}\*NODE;

NODE insert(NODE root,int x){

    if(root==NULL){

        root=(NODE)malloc(sizeof(struct node));

        root->val=x;

        root->left=root->right=NULL;

    }

    else if(x>root->val)

        root->right=insert(root->right,x);

    else if(x<root->val)

        root->left=insert(root->left,x);

    else{

        printf("Duplicate node\n");

        exit(0);

    }

    return(root);

}

void inorder(NODE cur){

    if(cur){

        inorder(cur->left);

        printf("%4d",cur->val);

        inorder(cur->right);

    }

}

int height(NODE cur){

    if (cur == NULL)

        return -1;

    else

        return MAX(height(cur->left),height(cur->right))+1;

}

void balancefactor(NODE cur){

    static int x;

    if(cur){

        balancefactor(cur->left);

        x = height(cur->left)-height(cur->right);

        printf("\nNode %d has a balance factor of %d",cur->val,x);

        balancefactor(cur->right);

    }

}

int main(){

    NODE root = NULL;

    int ch,x;

    do{

        printf("\n1.Enter element(no duplicates)  2. Print inorder  3. Show balance factor  4.Exit    Enter choice : ");

        scanf("%d",&ch);

        switch (ch){

            case 1 : printf("Enter element : ");

                     scanf("%d",&x);

                     root = insert(root,x);

                     break;

            case 2 : printf("\nInorder traversal is : ");

                     inorder(root);

                     break;

            case 3 : balancefactor(root);

                     break;

            case 4 : break;

            default:

                break;

        }

    }while(ch != 4);

    return 0;

}

**OUTPUT:**

1. **Write a program to create the AVL tree by iterative insertion.**

**CODE:**

**OUTPUT:**

**THE END**