**VARUN KUMAR**

**2K19-IT-140**

17: Write a Program to check weather given tree is Binary Search Tree or not

**#include<bits/stdc++.h>**

**using namespace std;**

**class TreeNode**

**{**

**public:**

**int val;**

**TreeNode\* left;**

**TreeNode\* right;**

**TreeNode(int val)**

**{**

**this->val = val;**

**this->left = NULL;**

**this->right = NULL;**

**}**

**};**

**bool Check\_BST(TreeNode\* TreeNode, int min, int max)**

**{**

**if (TreeNode==NULL)**

**return 1;**

**if (TreeNode->val < min || TreeNode->val > max)**

**return 0;**

**return**

**Check\_BST(TreeNode->left, min, TreeNode->val-1);**

**Check\_BST(TreeNode->right, TreeNode->val+1, max);**

**}**

**bool Is\_BST(TreeNode\* TreeNode)**

**{**

**return(Check\_BST(TreeNode, INT\_MIN, INT\_MAX));**

**}**

**int main()**

**{**

**TreeNode \*root = new TreeNode(5);**

**root->left = new TreeNode(4);**

**root->right = new TreeNode(10);**

**root->right->left = new TreeNode(16);**

**root->left->left = new TreeNode(6);**

**root->left->right = new TreeNode(2);**

**if(Is\_BST(root))**

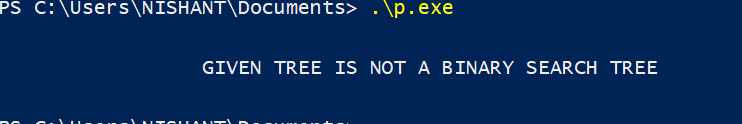
**cout<<"\n\n\t\t GIVEN TREE IS BINARY SEARCH TREE \n\n\n";**

**else**

**cout<<"\n\n\t\t GIVEN TREE IS NOT A BINARY SEARCH TREE \n\n\n";**

**return 0;**

**}**



# 18: Write a program to implement insertion in AVL tree.

**#include<bits/stdc++.h>**

**using namespace std;**

**class TreeNode**

**{**

**public:**

**int val;**

**TreeNode \*left;**

**TreeNode \*right;**

**int height;**

**};**

**int height(TreeNode \*cur)**

**{**

**if (cur == NULL)**

**return 0;**

**return cur->height;**

**}**

**TreeNode\* newNode(int val)**

**{**

**TreeNode\* root = new TreeNode();**

**root->val = val;**

**root->left = NULL;**

**root->right = NULL;**

**root->height = 1;**

**return(root);**

**}**

**TreeNode \*Rotate\_right(TreeNode \*y)**

**{**

**TreeNode \*x = y->left;**

**TreeNode \*T2 = x->right;**

**x->right = y;**

**y->left = T2;**

**y->height = max(height(y->left),**

**height(y->right)) + 1;**

**x->height = max(height(x->left),**

**height(x->right)) + 1;**

**return x;**

**}**

**TreeNode \*Rotate\_left(TreeNode \*x)**

**{**

**TreeNode \*y = x->right;**

**TreeNode \*T2 = y->left;**

**y->left = x;**

**x->right = T2;**

**x->height = max(height(x->left),**

**height(x->right)) + 1;**

**y->height = max(height(y->left),**

**height(y->right)) + 1;**

**return y;**

**}**

**int Balance\_Tree(TreeNode \*cur)**

**{**

**if (cur == NULL)**

**return 0;**

**return height(cur->left) - height(cur->right);**

**}**

**TreeNode\* Insert\_AVL(TreeNode\* TreeNode, int val)**

**{**

**if (TreeNode == NULL)**

**return(newNode(val));**

**if (val < TreeNode->val)**

**TreeNode->left = Insert\_AVL(TreeNode->left, val);**

**else if (val > TreeNode->val)**

**TreeNode->right = Insert\_AVL(TreeNode->right, val);**

**else**

**return TreeNode;**

**TreeNode->height = 1 + max(height(TreeNode->left),**

**height(TreeNode->right));**

**int balance = Balance\_Tree(TreeNode);**

**if (balance > 1 && val < TreeNode->left->val)**

**return Rotate\_right(TreeNode);**

**if (balance < -1 && val > TreeNode->right->val)**

**return Rotate\_left(TreeNode);**

**if (balance > 1 && val > TreeNode->left->val)**

**{**

**TreeNode->left = Rotate\_left(TreeNode->left);**

**return Rotate\_right(TreeNode);**

**}**

**if (balance < -1 && val < TreeNode->right->val)**

**{**

**TreeNode->right = Rotate\_right(TreeNode->right);**

**return Rotate\_left(TreeNode);**

**}**

**return TreeNode;**

**}**

**void PreOrder\_AVL(TreeNode \*root)**

**{**

**if(root != NULL)**

**{**

**cout << root->val <<" ";**

**PreOrder\_AVL(root->left);**

**PreOrder\_AVL(root->right);**

**}**

**}**

**int main()**

**{**

**TreeNode \*root = NULL;**

**root = Insert\_AVL(root, 10);**

**root = Insert\_AVL(root, 20);**

**root = Insert\_AVL(root, 30);**

**root = Insert\_AVL(root, 40);**

**root = Insert\_AVL(root, 50);**

**root = Insert\_AVL(root, 25);**

**/\* The constructed AVL Tree would be**

**30**

**/ \**

**20 40**

**/ \ \**

**10 25 50**

**\*/**

**cout << "PreOrder traversal of the "**

**"constructed AVL tree is : ----- \n\n";**

**PreOrder\_AVL(root);**

**return 0;**

**}**

