231501058 CS23231 – D a t a S t r u c t u r e s

**Ex. No.: 10**

**Implementation of AVL Tree**

**Date: 24/5/24**

**Write a function in C program to insert a ne w n ode with a g iven value into an AVL tre e . En sure that the tree remains balanced after insertion by pe rforming rotations if necessary . Repeat the above operation to de le te a n ode from AVL tree .**

**Algorithm:**

#include <stdio.h>

#include <malloc.h>

struct node {

int data;

struct node\* left;

struct node\* right;

int height;

};

struct node\* root = NULL;

struct node\* newnode;

int height(struct node\* N) {

if (N == NULL) return 0;

return N->height;

}

int max(int a, int b) {

return (a > b) ? a : b;

}

struct node\* rightRotate(struct node\* y) {

struct node\* x = y->left;

struct node\* 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;

}

struct node\* leftRotate(struct node\* x) {

struct node\* y = x->right;

struct node\* 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;



**Dept of Artificial Intelligence and Machine Learning** | **Rajalakshmi Engineering College**

. 38

231501058 CS23231 – D a t a S t r u c t u r e s

}

int getBalance(struct node\* N) {

if (N == NULL) return 0;

return height(N->left) - height(N->right);

}

struct node\* insert(struct node\* node, int ele) { if (node == NULL) {

newnode = (struct node\*)malloc(sizeof(struct node));

newnode->data = ele;

newnode->left = newnode->right = NULL;

newnode->height = 1;

return newnode;

}

if (ele < node->data) {

node->left = insert(node->left, ele);

} else if (ele > node->data) {

node->right = insert(node->right, ele);

} else {

return node;

}

node->height = 1 + max(height(node->left), height(node->right)); int balance = getBalance(node);

if (balance > 1 && ele < node->left->data) return rightRotate(node); if (balance < -1 && ele > node->right->data) return leftRotate(node); if (balance > 1 && ele > node->left->data) {

node->left = leftRotate(node->left);

return rightRotate(node);

}

if (balance < -1 && ele < node->right->data) { node->right = rightRotate(node->right); return leftRotate(node);

}

return node;

}

void inorder(struct node\* t) {

if (root == NULL) return;

if (t != NULL) {

inorder(t->left);

printf("%d ", t->data);

inorder(t->right);

}

}

void preorder(struct node\* t) {

if (root == NULL) return;

if (t != NULL) {



**Dept of Artificial Intelligence and Machine Learning** | **Rajalakshmi Engineering College**

. 39

231501058 CS23231 – D a t a S t r u c t u r e s

printf("%d ", t->data);

preorder(t->left);

preorder(t->right);

}

}

void postorder(struct node\* t) {

if (root == NULL) return;

if (t != NULL) {

postorder(t->left);

postorder(t->right);

printf("%d ", t->data);

}

}

int main() {

root = insert(root, 10);

root = insert(root, 20);

root = insert(root, 30);

root = insert(root, 40);

root = insert(root, 50);

root = insert(root, 25);

printf("Inorder Traversal: ");

inorder(root);

printf("\n");

printf("Preorder Traversal: ");

preorder(root);

printf("\n");

printf("Postorder Traversal: ");

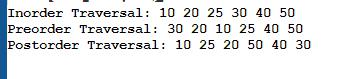
postorder(root);

printf("\n");

return 0;

}

**OUTPUT**



**Dept of Artificial Intelligence and Machine Learning** | **Rajalakshmi Engineering College**

. 40