***NAME : JANHAVI GATTANI***

***BATCH: 2***

***PRN : 12311291***

***ROLL NO : 37***

***LAB ASSIGNMENT 10***

**QUESTION:** **WAP to perform following operations on BST. a. Create b. Insert c. Delete d. Mirror Image e. Level wise Display f. Height of the tree g. Display Leaf Nodes.**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int data;

struct node \*left, \*right;

} Node;

Node\* createNode(int value) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = value;

newNode->left = newNode->right = NULL;

return newNode;

}

Node\* insert(Node\* root, int value) {

if (root == NULL)

return createNode(value);

if (value < root->data)

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

else if (value > root->data)

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

return root;

}

Node\* findMin(Node\* root) {

while (root->left != NULL)

root = root->left;

return root;

}

Node\* deleteNode(Node\* root, int key) {

if (root == NULL) return NULL;

if (key < root->data)

root->left = deleteNode(root->left, key);

else if (key > root->data)

root->right = deleteNode(root->right, key);

else {

if (root->left == NULL) {

Node\* temp = root->right;

free(root);

return temp;

}

else if (root->right == NULL) {

Node\* temp = root->left;

free(root);

return temp;

}

Node\* temp = findMin(root->right);

root->data = temp->data;

root->right = deleteNode(root->right, temp->data);

}

return root;

}

Node\* mirror(Node\* root) {

if (root == NULL) return NULL;

Node\* temp = root->left;

root->left = mirror(root->right);

root->right = mirror(temp);

return root;

}

int height(Node\* root) {

if (root == NULL) return 0;

int lh = height(root->left);

int rh = height(root->right);

return (lh > rh ? lh : rh) + 1;

}

void printLevel(Node\* root, int level) {

if (root == NULL) return;

if (level == 1)

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

else {

printLevel(root->left, level - 1);

printLevel(root->right, level - 1);

}

}

void levelOrder(Node\* root) {

int h = height(root);

printf("Level-wise Display:\n");

for (int i = 1; i <= h; i++) {

printf("Level %d: ", i);

printLevel(root, i);

printf("\n");

}

}

void printLeaves(Node\* root) {

if (root == NULL) return;

if (root->left == NULL && root->right == NULL)

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

printLeaves(root->left);

printLeaves(root->right);

}

void inorder(Node\* root) {

if (root == NULL) return;

inorder(root->left);

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

inorder(root->right);

}

void showTree(Node\* root) {

printf("\nCurrent Tree (Inorder): ");

inorder(root);

printf("\n");

levelOrder(root);

}

int main() {

Node\* root = NULL;

int choice, value;

while (1) {

printf("\n=== BST Operations ===\n");

printf("1. Insert\n2. Delete\n3. Mirror Tree\n4. Height of Tree\n5. Display Leaf Nodes\n6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

showTree(root);

break;

case 2:

printf("Enter value to delete: ");

scanf("%d", &value);

root = deleteNode(root, value);

showTree(root);

break;

case 3:

root = mirror(root);

printf("Mirror image created.\n");

showTree(root);

break;

case 4:

printf("Height of tree: %d\n", height(root));

break;

case 5:

printf("Leaf Nodes: ");

printLeaves(root);

printf("\n");

break;

case 6:

printf("Exiting...\n");

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

**OUTPUT:**



