**PRACTICAL NO -10**

**Program**:

#include <iostream>

#include <queue>

#include <iomanip>

#include <string>

using namespace std;

class node {

public:

int data;

int bal;

node \*left;

node \*right;

};

class Tree {

public:

node \*insert(node \*, node \*);

node \*rotateRight(node \*);

node \*rotateLeft(node \*);

void printTree(node \*);

int getHeight(node \*);

};

node \*Tree::insert(node \*root, node \*s) {

if (root == NULL)

return s;

if (s->data > root->data)

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

else

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

int leftHeight = getHeight(root->left);

int rightHeight = getHeight(root->right);

root->bal = leftHeight - rightHeight;

if (root->bal == 2) {

if (s->data < root->left->data)

return rotateRight(root);

else {

root->left = rotateLeft(root->left);

return rotateRight(root);

}

}

if (root->bal == -2) {

if (s->data > root->right->data)

return rotateLeft(root);

else {

root->right = rotateRight(root->right);

return rotateLeft(root);

}

}

return root;

}

node \*Tree::rotateRight(node \*root) {

node \*temp = root->left;

root->left = temp->right;

temp->right = root;

return temp;

}

node \*Tree::rotateLeft(node \*root) {

node \*temp = root->right;

root->right = temp->left;

temp->left = root;

return temp;

}

int Tree::getHeight(node \*root) {

if (!root) return 0;

return 1 + max(getHeight(root->left), getHeight(root->right));

}

void Tree::printTree(node\* root) {

if (!root) return;

int h = getHeight(root);

int maxWidth = 64;

queue<node\*> q;

q.push(root);

int level = 0;

while (!q.empty() && level < h) {

int levelSize = q.size();

int spacing = maxWidth / (levelSize + 1);

for (int i = 0; i < levelSize; ++i) {

node\* curr = q.front();

q.pop();

if (curr)

cout << setw(spacing) << "[" << curr->data << "]";

else

cout << setw(spacing) << " ";

if (curr) {

q.push(curr->left);

q.push(curr->right);

} else {

q.push(NULL);

q.push(NULL);

}

}

cout << "\n\n";

level++;

}

}

int main() {

char temp;

node \*root = NULL, \*s;

Tree t;

do {

s = new node;

s->bal = 0;

s->left = NULL;

s->right = NULL;

cout << "\nEnter node of tree: ";

cin >> s->data;

root = t.insert(root, s);

cout << "\nTree structure:\n\n";

t.printTree(root);

cout << "\nWANT TO ENTER MORE ELEMENTS (y/n): ";

cin >> temp;

} while (temp == 'y' || temp == 'Y');

return 0;

}

**Output** :

Enter node of tree: 14

Tree structure:

14

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 17

Tree structure:

14

17

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 11

Tree structure:

14

11 17

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 7

Tree structure:

14

11 17

7

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 53

Tree structure:

14

11 17

7 53

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 4

Tree structure:

14

7 17

4 11 53

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 13

Tree structure:

14

7 17

4 11 53

13

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 12

Tree structure:

14

7 17

4 12 53

11 13

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 8

Tree structure:

14

11 17

7 12 53

4 8 13

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 60

Tree structure:

14

11 53

7 12 17 60

4 8 13

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 9

Tree structure:

11

7 14

4 8 12 53

9 13 17 60

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 16

Tree structure:

11

7 14

4 8 12 53

9 13 17 60

16

WANT TO ENTER MORE ELEMENTS (y/n): y

Enter node of tree: 20

Tree structure:

11

7 14

4 8 12 53

9 13 17 60

16 20

WANT TO ENTER MORE ELEMENTS (y/n): n