Program:

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int data;

Node\* left;

Node\* right;

int leftThread; // leftThread=0 -> left pointer points to the inorder predecessor

int rightThread; // rightThread=0 -> right pointer points to the inorder successor

Node(int val){

this->data = val;

}

};

class DoubleThreadedBinaryTree{

private:

Node\* root;

public:

DoubleThreadedBinaryTree(){

// dummy Node with value as INT\_MAX

root = new Node(INT\_MAX);

root->left = root->right = root;

root->leftThread = 0;

root->rightThread = 1;

}

void insert(int data){

Node\* new\_node = new Node(data);

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

//Empty Tree

new\_node->left = root;

root->left = new\_node;

new\_node->leftThread = 0;

new\_node->rightThread = 0;

root->leftThread = 1;

new\_node->right = root;

return;

}

else{

Node\* current = root->left;

while(true){

if(current->data > data){

if(current->leftThread == 0 ){

// this is the last Node

new\_node->left = current->left;

current->left = new\_node;

new\_node->leftThread = current->leftThread;

new\_node->rightThread = 0;

current->leftThread = 1;

new\_node->right = current;

break;70

}

else{

current = current->left;

}

}

else{

if(current->rightThread == 0){

// this is the last Node

new\_node->right = current->right;

current->right = new\_node;

new\_node->rightThread = current->rightThread;

new\_node->leftThread = 0;

current->rightThread=1;

new\_node->left = current;

break;

}

else{

current = current->right;

}

}

}

}

}

Node\* findNextInorder(Node\* current){

if(current->rightThread == 0){

return current->right;

}

current = current->right;

while (current->leftThread != 0)

{

current = current->left;

}

return current;

}

void inorder(){

Node\* current = root->left;

while(current->leftThread == 1){

current = current->left;

}

while(current != root){

cout<<current->data<<" ";

current = findNextInorder(current);

}

cout<<"\n";

}

void preorder(){

Node\* current = root->left;

while(current != root){

cout<<current->data<<" ";

if(current->left != root && current->leftThread != 0)

current= current->left;

else if(current->rightThread == 1){

current = current->right;

}

else{

while (current->right != root && current->rightThread == 0)

{

current = current->right;

}

if(current->right == root)

break;71

else

{

current=current->right;

}

}

}

cout<<"\n";

}

};

int main(){

DoubleThreadedBinaryTree dtbt;

dtbt.insert(10);

dtbt.insert(1);

dtbt.insert(11);

dtbt.insert(5);

dtbt.insert(21);

dtbt.insert(17);

dtbt.insert(31);

dtbt.insert(100);

dtbt.inorder();

dtbt.preorder();

return 0;

}

**OUTPUT:**

1 5 10 11 17 21 31 100

10 1 5 11 21 17 31 100