# Practical 6 Source Code:-

#include <iostream>

struct Node {

int data; Node \*left; Node \*right;

bool isThreaded; // true if right pointer is a thread

Node(int val) : data(val), left(nullptr), right(nullptr), isThreaded(false) {}

};

class InOrderThreadedBinaryTree { private: Node \*root;

void insert(Node\*& root, int key) { if (root == nullptr) {

root = new Node(key);

return;

}

if (key < root->data) { insert(root->left, key);

} else {

if (root->isThreaded) { Node\* temp = root->right; root->right = new Node(key); root->right->right = temp; root->isThreaded = false;

} else {

insert(root->right, key);

}

}

}

void createThreaded(Node\* root) { if (root == nullptr) return;

createThreaded(root->left);

if (root->left != nullptr) { Node\* prev = root->left; while (prev->right != nullptr) { prev = prev->right;

}

prev->right = root;

prev->isThreaded = true;

}

createThreaded(root->right);

}

void preOrderTraverse(Node\* root) { if (root == nullptr) return;

std::cout << root->data << " ";

if (!root->isThreaded) {

preOrderTraverse(root->left);

}

if (root->right != nullptr && !root->isThreaded) { preOrderTraverse(root->right);

}

}

public:

InOrderThreadedBinaryTree() : root(nullptr) {}

void insert(int key) {

insert(root, key);

}

void createThreads() {

createThreaded(root);

}

void preOrder() {

preOrderTraverse(root);

}

};

int main() {

InOrderThreadedBinaryTree tree;

tree.insert(10); tree.insert(5); tree.insert(15); tree.insert(3); tree.insert(7); tree.insert(12); tree.insert(18);

// Create threads for in-order traversal tree.createThreads();

// Pre-order traversal of the threaded binary tree std::cout << "Pre-order traversal of the threaded binary tree:\n"; tree.preOrder();

return 0;

}

# Output:-

