IMPLEMENTATION:

/\* C++ program to convert a Binary Tree to

Threaded Tree \*/

#include <bits/stdc++.h>

using namespace std;

/\* Structure of a node in threaded binary tree \*/

struct Node

{

int key;

Node \*left, \*right;

// Used to indicate whether the right pointer

// is a normal right pointer or a pointer

// to inorder successor.

bool isThreaded;

};

// Converts tree with given root to threaded

// binary tree.

// This function returns rightmost child of

// root.

Node \*createThreaded(Node \*root)

{

// Base cases : Tree is empty or has single

// node

if (root == NULL)

return NULL;

if (root->left == NULL &&

root->right == NULL)

return root;

// Find predecessor if it exists

if (root->left != NULL)

{

// Find predecessor of root (Rightmost

// child in left subtree)

Node\* l = createThreaded(root->left);

// Link a thread from predecessor to

// root.

l->right = root;

l->isThreaded = true;

}

// If current node is rightmost child

if (root->right == NULL)

return root;

// Recur for right subtree.

return createThreaded(root->right);

}

// A utility function to find leftmost node

// in a binary tree rooted with 'root'.

// This function is used in inOrder()

Node \*leftMost(Node \*root)

{

while (root != NULL && root->left != NULL)

root = root->left;

return root;

}

// Function to do inorder traversal of a threadded

// binary tree

void inOrder(Node \*root)

{

if (root == NULL) return;

// Find the leftmost node in Binary Tree

Node \*cur = leftMost(root);

while (cur != NULL)

{

cout << cur->key << " ";

// If this Node is a thread Node, then go to

// inorder successor

if (cur->isThreaded)

cur = cur->right;

else // Else go to the leftmost child in right subtree

cur = leftMost(cur->right);

}

}

// A utility function to create a new node

Node \*newNode(int key)

{

Node \*temp = new Node;

temp->left = temp->right = NULL;

temp->key = key;

return temp;

}

// Driver program to test above functions

int main()

{

/\* 1

/ \

2 3

/ \ / \

4 5 6 7 \*/

Node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

root->right->left = newNode(6);

root->right->right = newNode(7);

createThreaded(root);

cout << "Inorder traversal of created "

"threaded tree is\n";

inOrder(root);

return 0;

}