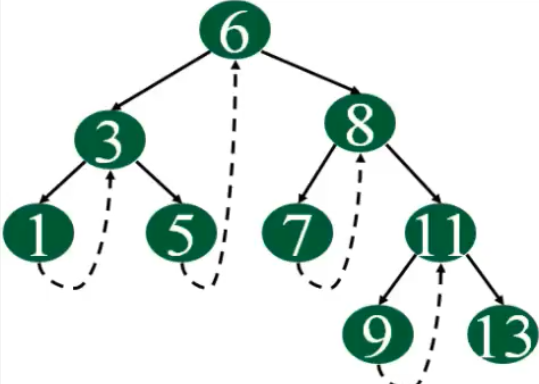
/\* C++ program to convert a Binary Tree to Threaded Tree \*/

#include <iostream.h>

using namespace std;

struct Node ****

{

    int key;

    Node \*left, \*right;

    bool isThreaded;

|  |  |  |
| --- | --- | --- |
| Null | 1 | Null |

\*root

|  |  |  |
| --- | --- | --- |
| Null | 2 | Null |

};

**Node \*createThreaded(Node \*root)**

**{**

**if (root == NULL)**

**return NULL;**

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

**return root;**

**if (root->left != NULL)**

**{**

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

**l->right = root;**

**l->isThreaded = true;**

**}**

**if (root->right == NULL)**

**return root;**

**return createThreaded(root->right);**

**}**

**Node \*leftMost(Node \*root)**

**{**

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

**root = root->left;**

**return root;**

**}**

**void inOrder(Node \*root)**

**{**

**if (root == NULL) return;**

**Node \*cur = leftMost(root);**

**while (cur != NULL)**

**{**

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

**if (cur->isThreaded)**

**cur = cur->right;**

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

**//cur = leftMost(cur->right);**

**}**

**}**

**Node \*newNode(int key)2**

**{**

**Node \*temp = new Node;**

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

**temp->key = key;**

**return temp;**

**}**

**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;**

**}**

/\* 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;

};

// Helper function to put the Nodes in inorder into queue

void populateQueue(Node\* root, std::queue<Node\*>\* q)

{

    if (root == NULL)

        return;

    if (root->left)

        populateQueue(root->left, q);

    q->push(root);

    if (root->right)

        populateQueue(root->right, q);

}

// Function to traverse queue, and make tree threaded

void createThreadedUtil(Node\* root, std::queue<Node\*>\* q)

{

    if (root == NULL)

        return;

    if (root->left)

        createThreadedUtil(root->left, q);

    q->pop();

    if (root->right)

        createThreadedUtil(root->right, q);

    // If right pointer is NULL, link it to the

    // inorder successor and set 'isThreaded' bit.

    else {

        root->right = q->front();

        root->isThreaded = true;

    }

}

// This function uses populateQueue() and

// createThreadedUtil() to convert a given binary tree

// to threaded tree.

void createThreaded(Node\* root)

{

    // Create a queue to store inorder traversal

    std::queue<Node\*> q;

    // Store inorder traversal in queue

    populateQueue(root, &q);

    // Link NULL right pointers to inorder successor

    createThreadedUtil(root, &q);

}

// 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 threaded 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;

}

**Dear Jijaji and Tanubala,**

**You went for the treatment..and**

**We were eating, but felt no taste to the food..**

**We were sleeping, but with nightmares..**

**We were doing routine stuff, but half-heartedly..**

**It was a tough time for all of us…really.**

**As if life had stopped at once..**

**But with God’s grace and blessings, now you both are back home safely,**

**no more worries!!**

**Everything will be back in shape, Life will be beautiful again,,**

**Welcome back!!**

**Have a speedy, complete recovery and may God bless you both**

**Always!!!!**