Convert given binary tree into threaded binary tree. Analyze time and space complexity

of the algorithm.

\*/

#include <bits/stdc++.h>

using namespace std;

struct Node

{

int value;

Node \*left, \*right;

bool rightThread;

};

Node \*convert(Node \*root)

{

if (root == NULL)

{

return NULL;

}

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

{

return root;

}

if (root->left != NULL)

{

Node\* a = convert(root->left);

a->right = root;

a->rightThread = true;

}

if (root->right == NULL)

{

return root;

}

return convert(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 \*current = leftmost(root);

while (current != NULL)

{

cout << current->value << " ";

if (current->rightThread)

{

current = current->right;

}

else

{

current = leftmost(current->right);

}

}

}

Node \*newNode(int value)

{

Node \*temp = new Node;

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

temp->value = value;

return temp;

}

int main()

{

int rt,l1,l2,l3,r1,r2,r3;

cout<<"\nEnter First Node ";

cin>>rt;

Node\* root = newNode(rt);

cout<<"\nEnter Left Child of "<<rt<<" ";

cin>>l1;

root->left = newNode(l1);

cout<<"\nEnter Right Child of "<<rt<<" ";

cin>>r1;

root->right = newNode(r1);

cout<<"\nEnter Left Child of "<<l1<<" ";

cin>>l2;

root->left->left = newNode(l2);

cout<<"\nEnter Right Child of "<<r1<<" ";

cin>>r2;

root->left->right = newNode(r2);

cout<<"\nEnter Left Child of "<<l2<<" ";

cin>>l3;

root->right->left = newNode(l3);

cout<<"\nEnter Left Child of "<<r2<<" ";

cin>>r3;

root->right->right = newNode(r3);

convert(root);

cout<<"InOrder Traversal of Created Threaded Binary Tree is\n";

inorder(root);

cout<<endl;

return 0;

}