#include<iostream>

#include<omp.h>

#include<stack>

#include<queue>

using namespace std;

struct TreeNode{

int val;

TreeNode\* left;

TreeNode\* right;

TreeNode(int x) : val(x), left(NULL), right(NULL) {}

};

void pBFS(TreeNode\* root){

queue<TreeNode\*> q;

q.push(root);

while(!q.empty()){

int qs = q.size();

#pragma omp parallel for

for(int i = 0; i < qs; i++){

TreeNode\* node;

#pragma omp critical

{

node = q.front();

cout << node->val << " ";

q.pop();

if(node->left) q.push(node->left);

if(node->right) q.push(node->right);

}

}

}

}

void pDFS(TreeNode\* root){

stack<TreeNode\*> s;

s.push(root);

while(!s.empty()){

int ss = s.size();

#pragma omp parallel for

for(int i = 0; i < ss; i++){

TreeNode\* node;

#pragma omp critical

{

node = s.top();

cout << node->val << " ";

s.pop();

if(node->right) s.push(node->right);

if(node->left) s.push(node->left);

}

}

}

}

int main(){

// Construct Tree

TreeNode\* tree = new TreeNode(1);

tree->left = new TreeNode(2);

tree->right = new TreeNode(3);

tree->left->left = new TreeNode(4);

tree->left->right = new TreeNode(5);

tree->right->left = new TreeNode(6);

tree->right->right = new TreeNode(7);

/\*

Our Tree Looks like this:

1

2 3

4 5 6 7

\*/

cout << "Parallel BFS: ";

pBFS(tree);

cout << "\n";

cout << "Parallel DFS: ";

pDFS(tree);

}