/\*

// specifies the number of threads to be used by OpenMP parallel regions in the program.

// OMP\_NUM\_THREADS : environment variable

// OMP\_NUM\_THREADS=4 : 4 indicates that the program intends to utilize 4 threads for parallel execution.

export OMP\_NUM\_THREADS=4

g++ -fopenmp bubblesort.cpp

./a.out

\*/

#include <omp.h>

#include<bits/stdc++.h>

using namespace std;

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

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

};

void bfs(TreeNode\* root) {

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

TreeNode\* node = q.front();

q.pop();

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

if (node->left) {

q.push(node->left);

}

if (node->right) {

q.push(node->right);

}

}

}

void dfs(TreeNode\* root) {

stack<TreeNode\*> s;

s.push(root);

while (!s.empty()) {

TreeNode\* node = s.top();

s.pop();

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

if (node->right) {

s.push(node->right);

}

if (node->left) {

s.push(node->left);

}

}

}

void parallel\_bfs(TreeNode\* root) {

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

int qSize = q.size();

#pragma omp parallel for

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

TreeNode\* node;

#pragma omp critical

{

node = q.front();

q.pop();

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

}

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

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

}

}

}

void parallel\_dfs(TreeNode\* root) {

stack<TreeNode\*> s;

s.push(root);

while (!s.empty()) {

int sSize = s.size();

#pragma omp parallel for

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

TreeNode\* node;

#pragma omp critical

{

node = s.top();

s.pop();

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

}

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

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

}

}

}

int main() {

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

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

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

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

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

cout << "BFS traversal: ";

//auto start = chrono::high\_resolution\_clock::now();

//double start = omp\_get\_wtime();

bfs(root);

//auto end = chrono::high\_resolution\_clock::now();

//double end = omp\_get\_wtime();

// cout << "\nBFS took " << end-start << " seconds." << endl;

cout << endl;

cout << "Parallel BFS traversal: ";

//start = omp\_get\_wtime();

parallel\_bfs(root);

//start = omp\_get\_wtime();

//cout << "\nParallel BFS took " << end-start << " seconds." << endl;

cout << endl;

cout << "---------------------------------------------------------"<<endl;

cout << "DFS traversal: ";

//start = omp\_get\_wtime();

dfs(root);

//end = omp\_get\_wtime();

//cout << "\nDFS took " << end - start << " seconds." << endl;

cout << endl;

cout << "Parallel DFS traversal: ";

//start = omp\_get\_wtime();

parallel\_dfs(root);

//end = omp\_get\_wtime();

//cout << "\nParallel DFS took " << end - start << " seconds." << endl;

return 0;

}

// Time Complexity: O(N) - visits every node in the tree exactly once