Shrikrushna Zirape :

41283 (BE2)

LP5 : Assign1

```
#include <iostream>
#include <queue>
#include <stack>
#include <vector>
#include <omp.h>

using namespace std;

struct TreeNode {
    int val;
    TreeNode* parent;
    vector<TreeNode*> children;
    bool visited;
};
void process(TreeNode* node) {
    cout << node->val << " ";
}

void bfs(TreeNode* root) {
    queue<TreeNode*> q;
    q.push(root);

    #pragma omp parallel
```

```
{
    while (!q.empty()) {
        TreeNode* node = nullptr;

        #pragma omp critical
        {
            if (!q.empty()) {
                node = q.front();
                q.pop();
            }
        }

        if (node != nullptr) {
            process(node);

            #pragma omp for
            for (int i = 0; i < node->children.size(); i++) {
                TreeNode* child = node->children[i];

                #pragma omp critical
                {
                    if (!child->visited) {
                        child->visited = true;
                        q.push(child);
                    }
                }
```

```
            }
        }
    }
}

void dfs(TreeNode* root) {
    stack<TreeNode*> s;
    s.push(root);

    #pragma omp parallel
    {
        while (!s.empty()) {
            TreeNode* node = nullptr;

            #pragma omp critical
            {
                if (!s.empty()) {
                    node = s.top();
                    s.pop();
                }
            }
            if (node != nullptr && !node->visited) {
                process(node);
                node->visited = true;
```

```
            #pragma omp for
            for (int i = 0; i < node->children.size(); i++) {
                TreeNode* child = node->children[i];

                #pragma omp critical
                {
                    if (!child->visited) {
                        s.push(child);
                    }
                }
            }
        }
    }
}

int main() {
    TreeNode* root = new TreeNode{1, nullptr, {}, false};
    TreeNode* n2 = new TreeNode{2, root, {}, false};
    TreeNode* n3 = new TreeNode{3, root, {}, false};
    TreeNode* n4 = new TreeNode{4, n2, {}, false};
    TreeNode* n5 = new TreeNode{5, n2, {}, false};
    TreeNode* n6 = new TreeNode{6, n3, {}, false};
    TreeNode* n7 = new TreeNode{7, n3, {}, false};

    root->children = {n2, n3};
```

```
    n2->children = {n4, n5};
    n3->children = {n6, n7};

    cout << "BFS: ";
    bfs(root);
    cout << endl;

    n2->visited = false;
    n3->visited = false;
    n4->visited = false;
    n5->visited = false;
    n6->visited = false;
    n7->visited = false;
    cout << "DFS: ";
    dfs(root);
    cout << endl;

    return 0;
}
/*

OUTPUT :

BFS: 1 2 3 4 5 6 7
DFS: 1 3 7 6 2 5 4
*/
```