Assignment No.

```
#include <iostream>
#include <vector>
#include <queue>
#include <stack>
#include <omp.h>
using namespace std;

class Graph {
    int V;
    vector<vector<int>> adj;
public:
    Graph(int V) : V(V), adj(V) {}
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

//Parallel BFS:
    void parallelBFS(int start) {
        vector<bool> vis(V, false);
        queue<int> q; q.push(start);
        vis[start] = true;

        #pragma omp parallel
        while (!q.empty()) {
            int node;
            #pragma omp critical
            { node = q.front(); q.pop(); }
            cout << node << " ";
            #pragma omp for
            for (int n : adj[node]) if (!vis[n]) {
                vis[n] = true;
                q.push(n);
```

```
            }
        }
    }

//Parallel DFS:
    void parallelDFS(int start) {
        vector<bool> vis(V, false);
        stack<int> s;
        s.push(start);

        #pragma omp parallel
        while (!s.empty()) {
            int node;
            #pragma omp critical
            { node = s.top(); s.pop(); }
            if (!vis[node]) {
                vis[node] = true; cout << node << " ";
                #pragma omp for
                for (int n : adj[node]) {
                    if (!vis[n]) s.push(n);
                }
            }
        }
    }
};

int main() {
    int V, E, u, v, start;
    cout << "Vertices: "; cin >> V;
    Graph g(V);
    cout << "Edges: "; cin >> E;
    cout << "Enter edges:\n";
    while (E--) {
        cin >> u >> v; g.addEdge(u, v);
    }
    cout << "Start node: "; cin >> start;
    cout << "BFS: "; g.parallelBFS(start);
    cout << "\nDFS: "; g.parallelDFS(start);
}
```

********** OUTPUT **********

```
Vertices: 6
Edges: 7
Enter edges:
0 1
0 2
1 3
1 4
2 4
3 5
4 5
Start node: 0
BFS: 0 1 2 3 4 5
DFS: 0 2 4 5 3 1
```