Name :- Umakant Dodtalle
Roll N.o :- 4127
Practical N.o 1 : Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP.

Code:-

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

using namespace std;

const int MAX_NODES = 100;
vector<int> graph[MAX_NODES];

void parallelBFS(int start) {
   bool visited[MAX_NODES] = {false};
   queue<int> q;
   q.push(start);
   visited[start] = true;

   while (!q.empty()) {
      int current = q.front();
      q.pop();
#pragma omp parallel for
      for (int i = 0; i < graph[current].size(); ++i) {
         int neighbor = graph[current][i];
#pragma omp critical
         {
            if (!visited[neighbor]) {
               q.push(neighbor);
               visited[neighbor] = true;
            }
         }
      }
   }

   cout << "BFS Visited Nodes: ";
   for (int i = 0; i < MAX_NODES; ++i) {
      if (visited[i]) {
            cout << i << " ";
        }
    }
    cout << endl;
}

void parallelDFS(int start, bool visited[]) {
    visited[start] = true;
#pragma omp parallel for
    for (int i = 0; i < graph[start].size(); ++i) {
        int neighbor = graph[start][i];
        if (!visited[neighbor]) {
            parallelDFS(neighbor, visited);
        }
    }
}

int main() {
    graph[0] = {1, 2};
    graph[1] = {0, 3, 4};
    graph[2] = {0, 5, 6};
    graph[3] = {1};
    graph[4] = {1};
    graph[5] = {2};
    graph[6] = {2};

    int start_node = 0;
    parallelBFS(start_node);

    bool visited[MAX_NODES] = {false};
    parallelDFS(start_node, visited);

    cout << "DFS Visited Nodes: ";
    for (int i = 0; i < MAX_NODES; ++i) {
        if (visited[i]) {
            cout << i << " ";
        }
    }
    cout << endl;

    return 0;
}
```

Output :-

```
PS E:\HPC> cd "e:\HPC\" ; if ($?) { g++ HPC1.cpp -o HPC1 } ; if ($?) { .\HPC1 }
BFS Visited Nodes: 0 1 2 3 4 5 6
DFS Visited Nodes: 0 1 2 3 4 5 6
```