Name: Snehal Jayprakash Borji
UID: 2023510008
Course: F.Y.M.C.A.
Subject: DS

Practical 06

Aim: Graph implementation: perform BFS operation

Problem Statement:

Input :Graph
Output : find the cycle is there in the graph starting from a vertex x, x is taken input from the user.

Coding:

Prac6.cpp

```cpp
include <iostream>

#include <vector>

#include <queue>

#include <unordered_set>

using namespace std;

class Graph {

public:

   int vertices;

   vector<vector<int>> adjList;



   Graph(int v) : vertices(v), adjList(v) {}
```

```
    void addEdge(int u, int v) {
        adjList[u].push_back(v);
        adjList[v].push_back(u); // Assuming an undirected graph
    }

    bool isCyclic(int startVertex) {
        vector<bool> visited(vertices, false);
        queue<pair<int, int>> q;  // Queue to perform BFS
        unordered_set<int> parentSet; // To keep track of parent vertices

        q.push({startVertex, -1}); // Starting vertex and its parent (no parent initially)

        while (!q.empty()) {
            int currentVertex = q.front().first;
            int parentVertex = q.front().second;
            q.pop();

            visited[currentVertex] = true;
```

```
            for (int neighbor : adjList[currentVertex]) {
                if (!visited[neighbor]) {
                    q.push({neighbor, currentVertex});
                } else if (neighbor != parentVertex) {
                    // If the neighbor is visited and not the parent, there
is a cycle
                    return true;
                }
            }
        }

        return false;
    }
};

int main() {
    int vertices, edges;
    cout << "Enter the number of vertices: ";
    cin >> vertices;
```

```
    Graph graph(vertices);

    cout << "Enter the number of edges: ";
    cin >> edges;

    cout << "Enter the edges (u v):" << endl;
    for (int i = 0; i < edges; ++i) {
        int u, v;
        cin >> u >> v;
        graph.addEdge(u, v);
    }

    int startVertex;
    cout << "Enter the starting vertex for cycle detection: ";
    cin >> startVertex;

    if (graph.isCyclic(startVertex)) {
        cout << "The graph contains a cycle starting from vertex " <<
startVertex << "." << endl;
```

```
    } else {

        cout << "No cycle found starting from vertex " << startVertex <<
"." << endl;

    }



    return 0;

}

#
```

OUTPUT :