Lab Exercise: Graph Class with BFS, DFS, and Graph Properties

---

Objective:

- Implement a graph using a class with an adjacency list.
- Perform Breadth-First Search (BFS) and Depth-First Search (DFS).
- Compute degrees, adjacent vertices, in-degree and out-degree.
- Handle both directed and undirected graphs.

Tasks:

1. Write a `Graph` class with the following members:
2. Constructor with number of vertices and a directed/undirected flag.
3. `add_edge(int u, int v)`: Adds an edge from `u` to `v`. If undirected, also add reverse edge.
4. `dfs(int v, vector<bool>& visited)`: Performs DFS from vertex `v`.
5. `full_dfs()`: Calls DFS from all unvisited vertices to handle disconnected graphs.
6. `bfs(int v, vector<bool>& visited)`: Performs BFS starting from vertex `v`.
7. `full_bfs()`: Calls BFS from all unvisited vertices to handle disconnected graphs.
8. `print_degrees()`: Prints the degree of each vertex.
9. `get_adjacent_vertices(int v)`: Returns all adjacent vertices of `v`.
10. `get_indegree(int v)`: Returns the in-degree of `v` (only meaningful for directed graphs).
11. `get_outdegree(int v)`: Returns the out-degree of `v`.
12. `print_indegree_outdegree()`: Prints in-degree and out-degree for all vertices (for directed graphs only).

Input Format:

- First line: `V E D`, where `V` = number of vertices, `E` = number of edges, `D` = 1 for directed, 0 for undirected.
- Next `E` lines: each contains two integers `u` and `v`, representing an edge.
- Last line: the start node for BFS/DFS.

Sample Input:

```
6 5 1
0 1
0 2
1 3
4 5
```

```
2 3
0
```

Expected Output:

```
DFS Traversal: 0 1 3 2 4 5
BFS Traversal: 0 1 2 3 4 5
Degrees:
Vertex 0: 2
Vertex 1: 1
Vertex 2: 1
Vertex 3: 0
Vertex 4: 1
Vertex 5: 0
Graph is not connected.
Number of connected components: 2
Adjacent vertices of 2: 3
In-degree of vertex 3: 2
Out-degree of vertex 0: 2
Vertex 0: In-degree = 0, Out-degree = 2
Vertex 1: In-degree = 1, Out-degree = 1
Vertex 2: In-degree = 1, Out-degree = 1
Vertex 3: In-degree = 2, Out-degree = 0
Vertex 4: In-degree = 0, Out-degree = 1
Vertex 5: In-degree = 1, Out-degree = 0
```

Bonus Tasks (Optional):

- Implement BFS using adjacency matrix instead of list.
- Implement iterative version of DFS.
- Add a method to find shortest path from source to all nodes using BFS.
- Use characters or strings as node labels.

Submission:

- Submit your `.cpp` file.
- Include sample input/output in comments.
- Bonus tasks should be clearly marked in your code.

Note: Ensure proper input validation and comment your code clearly.