WORKSHEET 2.1

Student Name: Megha Kaushal	**UID:** 21BCS10336
Branch: CSE	**Section/Group:** 601/A
Semester: 5th	**Date of Performance:** 12/09/2023
Subject Name: Advance Programming Lab-1	**Subject Code:** 21CSP-314

Aim: Implementing Graph Traversal (Visiting all the nodes of the graph).

Pseudo Code 1: Breadth First Search: Shortest Reach

Step 1. Define a Function "bfs":

- Create an empty list called "res" to store the results.
- Create an empty list called "marked" to keep track of which nodes have been visited.
- Create an empty list called "adj" to represent the graph structure.

Step 2. Take a list of edges and create an "adjacency list" from it, which shows which nodes are connected to each other.

Step 3. Initialize "marked" with all nodes as "unvisited."

Step 4. Create a Queue and Start BFS:

- Start BFS (a way of exploring a graph) from a given starting node "s."
- Use a queue to keep track of nodes to explore.
- Initialize an empty map called "res2" to store the distances.

Step 5. While there are nodes to explore in the queue:

- Take the front node from the queue.
- If the distance to this node is greater than 0, store it in "res2" for this node.
- Explore neighbors of this node and add them to the queue if they haven't been visited yet.

Steps 6. Fill the Result:

- Create a list of results called "res."
- For each node from 1 to "n" (excluding the starting node "s"):
    - If the node was visited, store its distance in "res."
    - If the node wasn't visited, store -1 in "res" to indicate that it couldn't be reached.

WORKSHEET 2.1

Student Name: Sonu Kumar
UID: 21BCS10550
Branch: CSE
Section/Group: 601/A
Semester: 5th
Date of Performance: 12/09/2023
Subject Name: Advance Programming Lab-1
Subject Code: 21CSP-314

Aim: Implementing Graph Traversal (Visiting all the nodes of the graph).

Pseudo Code 1: Breadth First Search: Shortest Reach

```
Step 1. Define a Function "bfs":
    • Create an empty list called "res" to store the results.
    • Create an empty list called "marked" to keep track of which nodes have been visited.
    • Create an empty list called "adj" to represent the graph structure.
Step 2. Take a list of edges and create an "adjacency list" from it, which shows which nodes are connected to each other.
Step 3. Initialize "marked" with all nodes as "unvisited."
Step 4. Create a Queue and Start BFS:
    • Start BFS (a way of exploring a graph) from a given starting node "s."
    • Use a queue to keep track of nodes to explore.
    • Initialize an empty map called "res2" to store the distances.
Step 5. While there are nodes to explore in the queue:
    • Take the front node from the queue.
    • If the distance to this node is greater than 0, store it in "res2" for this node.
    • Explore neighbors of this node and add them to the queue if they haven't been visited yet.
Steps 6. Fill the Result:
    • Create a list of results called "res."
    • For each node from 1 to "n" (excluding the starting node "s"):
        • If the node was visited, store its distance in "res."
        • If the node wasn't visited, store -1 in "res" to indicate that it couldn't be reached.
```