DAY 14:

ASSIGNMENT 1:

Task 5: Breadth-First Search (BFS) Implementation

For a given undirected graph, implement BFS to traverse the graph starting from a given node and print each node in the order it is visited.

ANSWER:

```
package day14;
import java.util.*;
	class Graph {
	    private int vertices; // Number of vertices
	    private LinkedList<Integer> adjList[]; // Adjacency List

	    // Constructor
	    Graph(int v) {
	        vertices = v;
	        adjList = new LinkedList[v];
	        for (int i = 0; i < v; ++i) {
	            adjList[i] = new LinkedList();
	        }
	    }

	    // Method to add an edge to the graph
	    void addEdge(int v, int w) {
	        adjList[v].add(w); // Add w to v's list
	        adjList[w].add(v); // Since the graph is undirected, also add v to
w's list
	    }

	    // BFS traversal from a given source s
	    void BFS(int s) {
	        // Mark all the vertices as not visited (by default set as false)
	        boolean visited[] = new boolean[vertices];

	        // Create a queue for BFS
	        LinkedList<Integer> queue = new LinkedList<>();

	        // Mark the current node as visited and enqueue it
	        visited[s] = true;
	        queue.add(s);

	        while (queue.size() != 0) {
	            // Dequeue a vertex from queue and print it
	            s = queue.poll();
	            System.out.print(s + " ");

	            // Get all adjacent vertices of the dequeued vertex s
	            // If an adjacent has not been visited, then mark it visited
and enqueue it
	            Iterator<Integer> i = adjList[s].listIterator();
	            while (i.hasNext()) {
	                int n = i.next();
	                if (!visited[n]) {
```

```
                        visited[n] = true;
                        queue.add(n);
                    }
                }
            }
        }

        // Driver method to test the BFS method
        public static void main(String args[]) {
            Graph g = new Graph(6);

            g.addEdge(0, 1);
            g.addEdge(0, 2);
            g.addEdge(1, 3);
            g.addEdge(1, 4);
            g.addEdge(2, 4);
            g.addEdge(3, 4);
            g.addEdge(3, 5);
            g.addEdge(4, 5);

            System.out.println("Breadth First Traversal starting from vertex
0:");

            g.BFS(0);
        }
    }
```

Explanation:

1. Graph Class:

- The Graph class has an integer vertices to represent the number of vertices and an array of LinkedList<Integer> to represent the adjacency list of the graph.

2. Constructor:

- Initializes the adjacency list for each vertex.

3. addEdge Method:

- Adds an edge between vertex v and vertex w in an undirected graph by updating the adjacency lists of both vertices.

4. BFS Method:

- Takes a starting vertex s.
- Uses a boolean array visited to keep track of visited vertices.
- Uses a queue to manage the BFS traversal.
- Marks the starting vertex as visited and enqueues it.

- Dequeues a vertex, prints it, and enqueues all its adjacent vertices that haven't been visited yet.

5. Main Method:

- Creates a graph with 6 vertices.
- Adds edges to the graph.
- Calls the BFS method starting from vertex 0 and prints the vertices in the order they are visited.