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.

1. CODE OF IMPLEMENTATION

```
Package com.Day14
import java.util.*;
public class BFSUndirectedGraph {
    private int numVertices; // Number of vertices in the graph
    private LinkedList<Integer> adjList[]; // Adjacency list
    public BFSUndirectedGraph(int vertices) {
        numVertices = vertices;
        adjList = new LinkedList[vertices];
        for (int i = 0; i < vertices; ++i)
            adjList[i] = new LinkedList<>();
    }
    void addEdge(int v, int w) {
        adjList[v].add(w); // Add w to v's list.
        adjList[w].add(v); // Add v to w's list since it's undirected
    }
    void BFS(int s) }
        boolean visited[] = new boolean[numVertices];
        LinkedList<Integer> queue = new LinkedList<>();
        visited[s] = true;
        queue.add(s);
        while (queue.size() != 0) {
            s = queue.poll();
            System.out.print(s + " ");
            Iterator<Integer> i = adjList[s].listIterator();
            while (i.hasNext()) {
                int n = i.next();
            if (!visited[n]) {
                visited[n] = true;
                queue.add(n);
            }           }       }    }

    public static void main(String args[]) {
        BFSUndirectedGraph g = new BFSUndirectedGraph(6);
        g.addEdge(0, 1);
        g.addEdge(0, 2);
        g.addEdge(1, 3);
        g.addEdge(2, 4);
        g.addEdge(3, 4);
        g.addEdge(3, 5);
        g.addEdge(4, 5);

        System.out.println("Following is Breadth First Traversal " +
                           "(starting from vertex 0):");
        g.BFS(0);    }       }
```

<u>Explanation:</u>

1. **Graph Initialization:**
   - numVertices holds the number of vertices.
   - adjList is an array of linked lists where each list represents the adjacency list of a vertex.
   - The constructor initializes the adjacency list for each vertex.
2. **Adding Edges**:
   - The addEdge method adds an edge between vertex v and vertex w.
   - Since the graph is undirected, the edge is added in both directions.
3. **BFS Method:**
   - An array visited is used to track which vertices have been visited.
   - A queue is used to explore nodes level by level.
   - Start by marking the source node s as visited and enqueue it.
   - While the queue is not empty, dequeue a vertex, print it, and enqueue all its unvisited neighbors after marking them as visited.

4. **Main Method:**
   - Creates a graph and adds some edges to represent an undirected graph.
   - Calls the BFS method starting from vertex 0 and prints the traversal order.