**Task 6: Depth-First Search (DFS) Recursive**

**Write a recursive DFS function for a given undirected graph. The function should visit every node and print it out.**

**import** java.util.\*;

**public** **class** GraphDFS {

**private** **int** V; // Number of vertices

**private** LinkedList<Integer> adj[]; // Adjacency list

// Constructor

GraphDFS(**int** v) {

V = v;

adj = **new** LinkedList[v];

**for** (**int** i = 0; i < v; ++i) {

adj[i] = **new** LinkedList<>();

}

}

// Function to add an edge to the graph

**void** addEdge(**int** v, **int** w) {

adj[v].add(w); // Add w to v's list

adj[w].add(v); // Since the graph is undirected, add v to w's list as well

}

// A function used by DFS

**void** DFSUtil(**int** v, **boolean** visited[]) {

// Mark the current node as visited and print it

visited[v] = **true**;

System.***out***.print(v + " ");

// Recur for all the vertices adjacent to this vertex

**for** (Integer n : adj[v]) {

**if** (!visited[n]) {

DFSUtil(n, visited);

}

}

}

// The function to do DFS traversal. It uses recursive DFSUtil()

**void** DFS(**int** v) {

// Mark all the vertices as not visited (set as false by default in Java)

**boolean** visited[] = **new** **boolean**[V];

// Call the recursive helper function to print DFS traversal starting from v

DFSUtil(v, visited);

}

**public** **static** **void** main(String args[]) {

GraphDFS g = **new** GraphDFS(5);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

g.addEdge(3, 4);

System.***out***.println("Following is Depth First Traversal starting from vertex 2");

g.DFS(2);

}

}