import java.io.\*;

import java.util.\*;

// This class represents a directed graph using adjacency

// list representation

class Graph {

// No. of vertices

private int V;

// Adjacency Lists

private LinkedList<Integer> adj[];

// Constructor

Graph(int v)

{

V = v;

adj = new LinkedList[v];

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

adj[i] = new LinkedList();

}

// Function to add an edge into the graph

void addEdge(int v, int w) { adj[v].add(w); }

// prints 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[V];

// Create a queue for BFS

LinkedList<Integer> queue

= new LinkedList<Integer>();

// 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 = adj[s].listIterator();

while (i.hasNext()) {

int n = i.next();

if (!visited[n]) {

visited[n] = true;

queue.add(n);

}

}

}

}

// Driver code

public static void main(String args[])

{

Graph g = new Graph(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

System.out.println(

"Following is Breadth First Traversal "

+ "(starting from vertex 2)");

g.BFS(2);

}

}