#include <stdio.h>

#define MAX 100

int adj[MAX][MAX]; // Adjacency matrix

int visited[MAX]; // Visited array

int queue[MAX]; // Queue for BFS

int front = -1, rear = -1;

// Enqueue operation

void enqueue(int vertex) {

if (rear == MAX - 1)

printf("Queue Overflow\n");

else {

if (front == -1) front = 0;

queue[++rear] = vertex;

}

}

// Dequeue operation

int dequeue() {

if (front == -1 || front > rear)

return -1;

else

return queue[front++];

}

// BFS traversal

void bfs(int start, int n) {

enqueue(start);

visited[start] = 1;

printf("BFS Traversal starting from vertex %d: ", start);

while (front <= rear) {

int current = dequeue();

printf("%d ", current);

for (int i = 0; i < n; i++) {

if (adj[current][i] == 1 && !visited[i]) {

enqueue(i);

visited[i] = 1;

}

}

}

printf("\n");

}

int main() {

int n, edges, u, v, start;

// Input: number of vertices

printf("Enter number of vertices: ");

scanf("%d", &n);

// Initialize adjacency matrix and visited array

for (int i = 0; i < n; i++) {

visited[i] = 0;

for (int j = 0; j < n; j++)

adj[i][j] = 0;

}

// Input: number of edges

printf("Enter number of edges: ");

scanf("%d", &edges);

// Input: edges

printf("Enter edges (u v):\n");

for (int i = 0; i < edges; i++) {

scanf("%d %d", &u, &v);

adj[u][v] = 1;

adj[v][u] = 1; // For undirected graph

}

// Input: starting vertex

printf("Enter starting vertex for BFS: ");

scanf("%d", &start);

// Perform BFS

bfs(start, n);

return 0;

}