#include <stdio.h>

#include <stdbool.h>

#define MAX\_NODES 100

#define MAX\_SIZE 6

// Queue structure

typedef struct {

int items[MAX\_SIZE];

int front, rear;

} Queue;

// Function prototypes for queue operations

Queue CreateQueue();

void enqueue(Queue \*q, int value);

int dequeue(Queue \*q);

void displayQueue(Queue \*q);

// Function prototypes for adjacency matrix operations

void addEdge(int src, int dest);

void displayAdjMatrix();

// Global variables

int adjMatrix[MAX\_NODES][MAX\_NODES] = {0};

int numNodes;

// BFS function

void BFS(int startNode) {

// Create a queue

Queue q = CreateQueue();

// Boolean array to mark visited nodes

bool visited[MAX\_NODES] = {false};

// Enqueue the start node

enqueue(&q, startNode);

visited[startNode] = true;

// Display the current node being visited

printf("BFS Traversal starting from node %d: ", startNode);

// Loop until the queue is empty

while (q.front != -1) {

// Dequeue a node from the queue

int currentNode = dequeue(&q);

printf("%d ", currentNode);

// Visit all adjacent nodes of the dequeued node

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

if (adjMatrix[currentNode][i] == 1 && !visited[i]) {

enqueue(&q, i);

visited[i] = true;

}

}

}

printf("\n");

}

int main() {

int numEdges, src, dest;

// Input the number of nodes

printf("Enter the number of nodes: ");

scanf("%d", &numNodes);

if (numNodes > MAX\_NODES) {

printf("Number of nodes exceeds the maximum limit.\n");

return 1;

}

// Input the number of edges

printf("Enter the number of edges: ");

scanf("%d", &numEdges);

// Input the edges (source destination)

printf("Enter the edges (source destination):\n");

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

scanf("%d %d", &src, &dest);

if (src < 0 || src >= numNodes || dest < 0 || dest >= numNodes) {

printf("Invalid edge. Node index out of bounds.\n");

return 1;

}

addEdge(src, dest);

}

// Display the adjacency matrix

printf("\n");

displayAdjMatrix();

// Input the start node for BFS traversal

int startNode;

printf("Enter the start node for BFS traversal: ");

scanf("%d", &startNode);

if (startNode < 0 || startNode >= numNodes) {

printf("Invalid start node. Node index out of bounds.\n");

return 1;

}

// Perform BFS traversal

BFS(startNode);

return 0;

}

// Function to create an empty queue

Queue CreateQueue() {

Queue q;

q.front = -1;

q.rear = -1;

return q;

}

// Function to enqueue an element into the queue

void enqueue(Queue \*q, int value) {

if ((q->rear + 1) % MAX\_SIZE == q->front) {

printf("Queue is full. Cannot enqueue.\n");

return;

}

if (q->front == -1)

q->front = 0;

q->rear = (q->rear + 1) % MAX\_SIZE;

q->items[q->rear] = value;

}

// Function to dequeue an element from the queue

int dequeue(Queue \*q) {

int value;

if (q->front == -1) {

printf("Queue is empty. Cannot dequeue.\n");

return -1;

}

value = q->items[q->front];

if (q->front == q->rear) {

q->front = -1;

q->rear = -1;

} else {

q->front = (q->front + 1) % MAX\_SIZE;

}

return value;

}

// Function to display the elements of the queue

void displayQueue(Queue \*q) {

if (q->front == -1) {

printf("Queue is empty.\n");

return;

}

printf("Queue elements: ");

int i = q->front;

do {

printf("%d ", q->items[i]);

i = (i + 1) % MAX\_SIZE;

} while (i != (q->rear + 1) % MAX\_SIZE);

printf("\n");

}

// Function to add an edge between two nodes in the adjacency matrix

void addEdge(int src, int dest) {

adjMatrix[src][dest] = 1;

}

// Function to display the adjacency matrix

void displayAdjMatrix() {

printf("Adjacency Matrix:\n");

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

for (int j = 0; j < numNodes; j++) {

printf("%d ", adjMatrix[i][j]);

}

printf("\n");

}

}