#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

struct Node {

int data;

struct Node\* next;

};

struct Graph {

int numVertices;

struct Node\*\* adjLists;

int\* visited;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

struct Graph\* createGraph(int numVertices) {

struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph));

graph->numVertices = numVertices;

graph->adjLists = (struct Node\*\*)malloc(numVertices \* sizeof(struct Node\*));

graph->visited = (int\*)malloc(numVertices \* sizeof(int));

int i;

for (i = 0; i < numVertices; i++) {

graph->adjLists[i] = NULL;

graph->visited[i] = 0;

}

return graph;

}

void addEdge(struct Graph\* graph, int src, int dest) {

struct Node\* newNode = createNode(dest);

newNode->next = graph->adjLists[src];

graph->adjLists[src] = newNode;

newNode = createNode(src);

newNode->next = graph->adjLists[dest];

graph->adjLists[dest] = newNode;

}

void printGraph(struct Graph\* graph) {

int v;

for (v = 0; v < graph->numVertices; v++) {

struct Node\* temp = graph->adjLists[v];

printf("Adjacency list of vertex %d\n", v);

while (temp) {

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

}

void BFS(struct Graph\* graph, int startVertex) {

int queue[MAX\_SIZE];

int front = -1, rear = -1;

graph->visited[startVertex] = 1;

queue[++rear] = startVertex;

while (front != rear) {

int currentVertex = queue[++front];

printf("%d ", currentVertex);

struct Node\* temp = graph->adjLists[currentVertex];

while (temp) {

int adjVertex = temp->data;

if (graph->visited[adjVertex] == 0) {

graph->visited[adjVertex] = 1;

queue[++rear] = adjVertex;

}

temp = temp->next;

}

}

}

int main() {

struct Graph\* graph = createGraph(6);

addEdge(graph, 0, 1);

addEdge(graph, 0, 2);

addEdge(graph, 1, 3);

addEdge(graph, 2, 3);

addEdge(graph, 2, 4);

addEdge(graph, 3, 4);

addEdge(graph, 3, 5);

printf("Graph:\n");

printGraph(graph);

printf("BFS Traversal: ");

BFS(graph, 0);

return 0;

}

