#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_VERTICES 100

struct Queue {

int front, rear, size;

unsigned capacity;

int\* array; };

struct Queue\* createQueue(unsigned capacity) {

struct Queue\* queue = (struct Queue\*)malloc(sizeof(struct Queue));

queue->capacity = capacity;

queue->front = queue->size = 0;

queue->rear = capacity - 1;

queue->array = (int\*)malloc(queue->capacity \* sizeof(int));

return queue;

}

bool isEmpty(struct Queue\* queue) {

return (queue->size == 0);

}

bool isFull(struct Queue\* queue) {

return (queue->size == queue->capacity);

}

void enqueue(struct Queue\* queue, int item) {

if (isFull(queue)) return;

queue->rear = (queue->rear + 1) % queue->capacity;

queue->array[queue->rear] = item;

queue->size++;

}

int dequeue(struct Queue\* queue) {

if (isEmpty(queue)) return -1;

int item = queue->array[queue->front];

queue->front = (queue->front + 1) % queue->capacity;

queue->size--;

return item; }

struct AdjListNode {

int dest;

struct AdjListNode\* next;

};

struct AdjList {

struct AdjListNode\* head;

};

struct Graph {

int V;

struct AdjList\* array;

};

struct AdjListNode\* newAdjListNode(int dest) {

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

newNode->dest = dest;

newNode->next = NULL;

return newNode;

}

struct Graph\* createGraph(int V) {

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

graph->V = V;

graph->array = (struct AdjList\*)malloc(V \* sizeof(struct AdjList));

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

graph->array[i].head = NULL;

}

return graph;

}

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

struct AdjListNode\* newNode = newAdjListNode(dest);

newNode->next = graph->array[src].head;

graph->array[src].head = newNode;

newNode = newAdjListNode(src);

newNode->next = graph->array[dest].head;

graph->array[dest].head = newNode;

}

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

bool\* visited = (bool\*)malloc(graph->V \* sizeof(bool));

for (int i = 0; i < graph->V; i++) {

visited[i] = false;

}

struct Queue\* queue = createQueue(graph->V);

visited[src] = true;

enqueue(queue, src);

printf("Breadth-First Traversal starting from vertex %d: ", src);

while (!isEmpty(queue)) {

int vertex = dequeue(queue);

printf("%d ", vertex);

struct AdjListNode\* currentNode = graph->array[vertex].head;

while (currentNode != NULL) {

int adjVertex = currentNode->dest;

if (!visited[adjVertex]) {

visited[adjVertex] = true;

enqueue(queue, adjVertex);

}

currentNode = currentNode->next; } }

printf("\n");

free(visited);

free(queue->array);

free(queue); }

int main() {

int V = 6;

struct Graph\* graph = createGraph(V);

addEdge(graph, 0, 1);

addEdge(graph, 1, 2);

addEdge(graph, 1, 3);

addEdge(graph, 1, 4);

addEdge(graph, 3, 4);

addEdge(graph, 3, 6);

int sourceVertex = 0;

BFS(graph, sourceVertex);

return 0; }