a) Write a program to traverse a graph using BFS method. Program:

#include <stdio.h> #include <stdlib.h> #include <stdbool.h>

#define MAX 100 struct Queue {

int items[MAX];

int front, rear;

};

void initQueue(struct Queue\* q) { q->front = -1;

q->rear = -1;

}

bool isEmpty(struct Queue\* q) { return q->front == -1;

}

void enqueue(struct Queue\* q, int value) { if (q->rear == MAX - 1)

return;

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

q->rear++;

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

}

int dequeue(struct Queue\* q) { if (isEmpty(q))

return -1;

int item = q->items[q->front]; if (q->front == q->rear) {

q->front = q->rear = -1;

} else {

q->front++;

}

return item;

}

struct Graph { int vertices;

int adjMatrix[MAX][MAX];

};

void initGraph(struct Graph\* g, int vertices) { g->vertices = vertices;

for (int i = 0; i < vertices; i++) { for (int j = 0; j < vertices; j++) {

g->adjMatrix[i][j] = 0;

}

}

}

void addEdge(struct Graph\* g, int u, int v) { g->adjMatrix[u][v] = 1;

g->adjMatrix[v][u] = 1;

}

void bfs(struct Graph\* g, int start) { bool visited[MAX] = {false}; struct Queue q;

initQueue(&q); visited[start] = true; enqueue(&q, start);

while (!isEmpty(&q)) {

int node = dequeue(&q); printf("%d ", node);

for (int i = 0; i < g->vertices; i++) {

if (g->adjMatrix[node][i] == 1 && !visited[i]) { visited[i] = true;

enqueue(&q, i);

}

}

}

}

int main() { struct Graph g;

initGraph(&g, 6);

addEdge(&g, 0, 1);

addEdge(&g, 0, 2);

addEdge(&g, 1, 3);

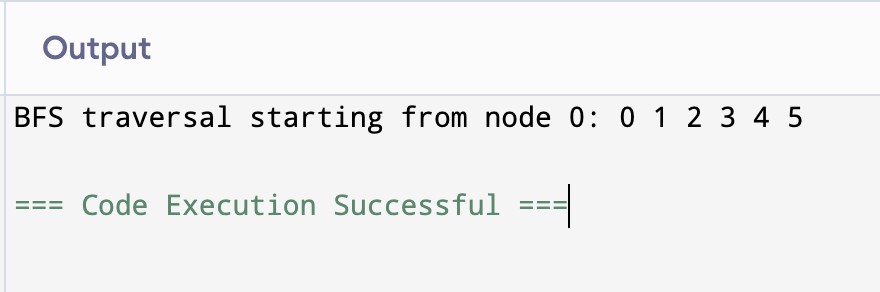
addEdge(&g, 1, 4);

addEdge(&g, 2, 5);

printf("BFS traversal starting from node 0: "); bfs(&g, 0);

return 0;

}



b) Write a program to check whether given graph is connected or not using DFS method.

Program:

#include <stdio.h> #include <stdlib.h> #include <stdbool.h>

#define MAX 100 struct Graph {

int vertices;

int adjMatrix[MAX][MAX];

};

void initGraph(struct Graph\* g, int vertices) { g->vertices = vertices;

for (int i = 0; i < vertices; i++) { for (int j = 0; j < vertices; j++) {

g->adjMatrix[i][j] = 0;

}

}

}

void addEdge(struct Graph\* g, int u, int v) { g->adjMatrix[u][v] = 1;

g->adjMatrix[v][u] = 1;

}

void dfs(struct Graph\* g, int vertex, bool visited[]) { visited[vertex] = true;

for (int i = 0; i < g->vertices; i++) {

if (g->adjMatrix[vertex][i] == 1 && !visited[i]) { dfs(g, i, visited);

}

}

}

bool isConnected(struct Graph\* g) { bool visited[MAX] = {false}; dfs(g, 0, visited);

for (int i = 0; i < g->vertices; i++) { if (!visited[i]) {

return false;

}

}

return true;

}

int main() { struct Graph g;

int vertices = 6; initGraph(&g, vertices);

addEdge(&g, 0, 1);

addEdge(&g, 0, 2);

addEdge(&g, 1, 3);

addEdge(&g, 1, 4);

addEdge(&g, 2, 5);

if (isConnected(&g)) {

printf("The graph is connected.\n");

} else {

printf("The graph is not connected.\n");

}

return 0;

}

