1. Graph code in c

main.c   Run   Output   Clear

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_VERTICES 100

// Structure to represent a graph
typedef struct {
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];
    int numVertices;
} Graph;

// Function to initialize a graph
void initGraph(Graph* g, int vertices) {
    g->numVertices = vertices;
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            g->adjMatrix[i][j] = 0; //
                Initialize all edges as 0 (no
                edge)
        }
    }
}

// Function to add an edge to the graph
void addEdge(Graph* g, int src, int dest) {
    if (src >= g->numVertices || dest >= g
        ->numVertices || src < 0 || dest < 0) {
        printf("Invalid edge!\n");
        return;
    }
    g->adjMatrix[src][dest] = 1;
    g->adjMatrix[dest][src] = 1; // Because the
        graph is undirected
}

// Function to print the graph's adjacency
    matrix
void printGraph(Graph* g) {
    for (int i = 0; i < g->numVertices; i++) {
        for (int j = 0; j < g->numVertices; j
            ++) {
            printf("%d ", g->adjMatrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    Graph g;
    int vertices = 4; // Number of vertices in
        the graph

    initGraph(&g, vertices);

    // Add some edges
    addEdge(&g, 0, 1);
    addEdge(&g, 0, 2);
    addEdge(&g, 1, 2);
    addEdge(&g, 1, 3);

    // Print the graph
    printf("Adjacency Matrix of the Graph:\n");
    printGraph(&g);

    return 0;
}
```

Output:
```
/tmp/zMfFgULw3W.o
Adjacency Matrix of the Graph:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0

=== Code Execution Successful ===
```