Data Structures and Algorithms Assignment 5: Graph

This assignment contains two parts: 5 MCQ and 3 Programming questions. The templates for Programming Questions 1-3 are given as separated files (Q1_template.c, Q2_template.c, and Q3_template.c). You must use them to implement your functions. The program contains a main() function, which will ask you to input the number of vertices and the two vertices for each edge in a graph. You need to submit your code to the https://www.hackerearth.com (Email invitation will be sent to your school account). You need to submit your code to the NTULearn (State your name and your lab group in your submission) as well. Considering April 10, 2024 is a public holiday, deadline for this assignment submission is April 16, 2024 (Tuesday) 11.59 pm.

Programming Question 1 (Graph Connection of Directed Graph): Based on the DFS algorithm, write a function, Connected(), to determine if an undirected graph is connected or not (i.e., an undirected graph is connected if there is a path from any vertex to any other vertex). Vertices ranged from 1 to |V|. You may assume the input graph is always valid (no duplicate or invalid link. etc.).

int Connected (Graph g);

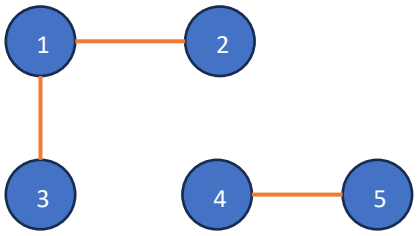
Herer Graph is defined as:

```
typedef struct _graph{
    int V;
    int E;
    int *visited;
    int **matrix;
}Graph;
```

where **matrix** is the adjacency matrix, and **visited** can be used to record the vertices that have been visited when conducting DFS. The function will return 1 if the graph is connected. Any other values will imply not being connected.

A sample input is given as follows:

```
5
1 2
1 3
4 5
a
```



where 5 is the number of vertices, each line after the first line and before “a” specifies an edge between the two vertices of that line. This input specifies the graph on the right. This graph is not connected.

Programming Question 2 (Graph Connection of Directed Graph): Based on the DFS algorithm, write a function Connected(), to determine if a directed graph is strongly connected or not (i.e., a directed graph is strongly connected if there is a path from any vertex to any other vertex). Vertices ranged from 1 to |V|. You may assume that the input graph is always valid (no duplicate or invalid link etc.). The function prototype is given as follows:

int Connected (Graph g);

Herer Graph is defined as:

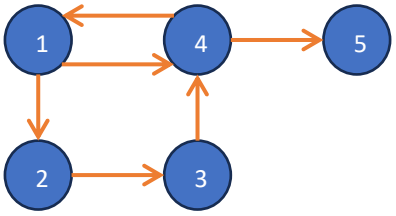
```
typedef struct _graph{
    int V;
    int E;
    int *visited;
    int **matrix;
}Graph;
```

where **matrix** is the adjacency matrix, and **visited** can be used to record the vertices that have been visited when conducting DFS.

The function will return 1 if the graph is strongly connected. Any other values will imply not being strongly connected.

A sample input is given as follows:

```
5
1 2
1 4
2 3
3 4
4 1
4 5
a
```



where 5 is the number of vertices, each line after the first line and before “a” specifies an edge from the first vertex to the second of that line. This input specifies the graph on the right. The graph is not strongly connected.

Programming Question 3 (Shortest Distance): Based on the BFS algorithm, write a function SD() to find the shortest distance from vertex v to vertex w, in an undirected graph. Vertices ranged from 1 to |V|. The distance is measured by the number of edges. If there is no path from v to w, then -1 is returned. You may assume that the input graph is always valid (no duplicate or any invalid link, etc.). The function prototype is given as follows:

int SD (Graph G, int v, int w);

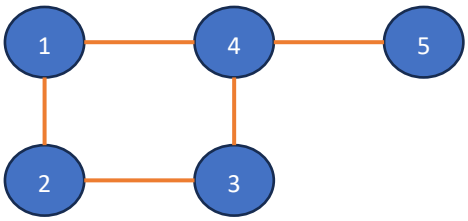
Herer Graph is defined as:

```
typedef struct _graph{
    int V;
    int E;
    int *visited;
    int **matrix;
}Graph;
```

where **matrix** is the adjacency matrix, and **visited** can be used to record the vertices that have been visited when conducting BFS.

A sample input is given as follows:

```
5
1 2
1 4
2 3
3 4
4 5
a
1 5
```



where 5 is the number of vertices, each line after the first line and before “a” specifies an edge between the two vertices of that line. This input specifies the graph on the right. The final line specifies the two vertices for which you are required to find the shortest distance. The shortest between 1 and 5 is 2.