Exercise 9. Graphs Representation

(graphs.c & graphs.h)

Input from file: graphs.in
Output on file: graphs.out

Graph has been widely used in the field of computer science. Many real-life problems are solvable via the approach of graph modeling. Programmers should know how to model/represent graphs on computers. With that, your task is to educate programmers on how graphs are represented on programs via creating a program that displays the graph representation given the edges using both the adjacency matrix and adjacency list data structure.

Input format:

The input contains several test cases, each representing the attributes of a graph. The input file begins with an integer m (1<=m <=10) representing the total number of test cases. The data for each case begins with an integer n (1<=n<=10) representing the total number of vertices. This is followed by an integer m (1 <= m <= (n*(n-1)/2)) representing the total number of edges. Followed by, k representing whether the graph is undirected (0) or directed (1). Then, it is followed by m lines of input data (edge description in the form of vertex1, vertex2).

Output format:

For each test case, output the test case number and the configuration of the representation of the graph on both data structure. Test cases are separated by a line.

Sample Input:

```
2
6
7
0
0, 1
0, 4
1, 2
1, 4
2, 3
3, 4
3, 5
6
7
1
0, 1
0, 4
1, 2
1, 4
2, 3
3, 4
3,5
```

Sample Output:

Case 1:

Adjacency Matrix Representation:

```
0 1 0 0 1 0
1 0 1 0 1 0
0 1 0 1 0 0
0 0 1 0 1 1
1 1 0 1 0 0
0 0 0 1 0 0
```

Adjacency List Representation:

[0] -> 1, 4,
[1] -> 0, 2, 4,
[2] -> 1, 3,
[3] -> 2, 4, 5,
[4] -> 0, 1, 3,
[5] -> 3,

Prepared by: ZO Arnejo
ICS, UPLB

Reference: ACM-ICPC format

Case 2:

Adjacency Matrix Representation:

```
0 1 0 0 1 0
0 0 1 0 1 0
0 0 0 1 0 0
0 0 0 0 1 1
0 0 0 0 0 0
0 0 0 0 0 0
```

Adjacency List Representation:

[0] -> 1, 4,
[1] -> 2, 4,
[2] -> 3,
[3] -> 4, 5,
[4] ->
[5] ->