**CSE 2321** **LAB 1** **AUTUMN 2016**

**Due: Electronic Submission 11:59 pm Sunday, October 30th**
**Hardcopy Materials: Start of class Monday, October 31st**

## Goal

Design and implement an algorithm to output the connected components of an undirected simple graph as well as the total number of components.

### Details

Input to your program consists of an integer *n* representing the number of vertices followed an adjacency matrix of *n* rows, each row with *n* entries.

Each entry of the *n* x *n* adjacency matrix, will contain a 1 if there is an edge between the corresponding vertices or a 0 if there is no edge.

Use the recursive depth first search algorithm, see slide 4 of the GraphTheoryAlgorithms slides, to traverse the graph and output the vertices of a component. Once the DFS is finished, determine if the DFS has visited every vertex. If not, repeat the DFS finding the next component until all components have been output. Lastly, output the total number of components.

**EXAMPLE 1:**
**Input (Adjacency Matrix Representation)**

```
5           // 5 vertices in graph.  There will be 5 rows of the matrix, each with 5 entries.
0 0 1 0 0   // Row 1 of Adjacency Matrix, vertices adjacent to vertex 1
0 0 1 0 0   // Row 2 of Adjacency Matrix, vertices adjacent to vertex 2
1 1 0 1 1   // Row 3 of Adjacency Matrix, vertices adjacent to vertex 3
0 0 1 0 1   // Row 4 of Adjacency Matrix, vertices adjacent to vertex 4
0 0 1 1 0   // Row 5 of Adjacency Matrix, vertices adjacent to vertex 5
```

**Output**

```
Component: 1 3 2 4 5
Total number of Components: 1
```

**EXAMPLE 2:**
**Input (Adjacency Matrix Representation)**

```
7
0 0 1 0 1 0 1
0 0 0 1 0 0 1
1 0 0 0 1 0 0
0 1 0 0 0 0 1
1 0 1 0 0 0 1
0 0 0 0 0 0 0
1 1 0 1 1 0 0
```

**Output**

```
Component: 1 3 5 7 2 4
```

Component: 6
Total Number of Components: 2

**EXAMPLE 3:**

**Input (Adjacency Matrix Representation)**

10
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0

**Output**

Component: 1
Component: 2
Component: 3
Component: 4
Component: 5
Component: 6 9
Component: 7
Component: 8
Component: 10
Total number of Components: 8

**Notes:**

- The depth first search should start at the first vertex of the graph which is represented by the first row of the adjacency matrix. Each vertex should be output no more than once.
- You may assume a max matrix of *n* = 12, that is a 12 x 12 matrix. The number of components could be between 1 and *n* inclusive.
- ***You may write your program in C, C++, or Java. Save your program in a file called lab1.* where * is the file extension. Do not use components, libraries, etc. for this program.***
- This program must run on the CSE stdlinux environment, if you do not know your username and/or password, please view information on the CSE website under Computing Services.
- Code that does not compile on stdlinux or has execution errors will be given a score of 0. ***It is your responsibility to make sure that your code compiles and runs correctly in the CSE stdlinux environment.***
- Your program should read from standard input. For instance, if the executable of your program is called lab1 you can run it on the file data1 by typing:

  lab1 < data1.

  You will need to create the data files. ***Your program should not be doing file I/O.***
- Use the submit command from the command line to submit your program, that is,

'**submit** submit-directory **lab1 lab1.***', where submit-directory is **c2321ai** for the MWF 12:40 section, **c2321ad** for the MWF 1:50 section, and * is the file extension. You should be in the folder/directory where your lab file is located when you type the submit command. ***It is your responsibility to confirm that your program submitted successfully.***

- Code should be commented and should contain a README header stating the student's name, email address, and instructions on how to compile and run the code.
- Source code should be submitted electronically by **11:59 Sunday, October 30th.** A late penalty of 10% **per hour** will be assessed, for instance, if a lab is submitted at 12:00 am or 12:59 am there is a 10% penalty.
- A printout of the source code along with screen shots of the output of the examples above along with 2 other examples of your choosing should be turned in at the **start of class on Monday, October 31st .**
- ***This programming assignment is to be done individually. You may not discuss the design, implementation, debugging and/or testing with other students although you may ask questions regarding the interpretation of the problem or the meaning of error messages on Piazza. DO NOT POST ANY CODE ON PIAZZA. In addition, do not look at or copy anyone else's code or any code from the Internet or other references.***

**References:**

- Adjacency Matrix: GraphTheoryAlgorithm Lecture PowerPoint Slides,
  also Textbook, pages 589-591
- Depth First Search: GraphTheoryAlgorithm Lecture PowerPoint Slides,
  also Textbook, pages 603-606
- 2-Dimensional Array: Section 6.7 from Software I/II Java Reference book – free online at http://proquest.safaribooksonline.com.proxy.lib.ohio-state.edu/book/programming/java/9781118087886/chapter-6-arrays-and-array-lists/navpoint-54#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODExMTgwODc4ODYlMkZuYXZwb2ludC02MCZxdWVyeT0