BITS Pilani, Pilani Campus
2nd Sem. 2017-18
CS F211 Data Structures & Algorithms
==========
Lab XI - (9th April to 14th April)
==========

Topics: Topological Sorting of Directed Acyclic Graphs

Exercise 1: Assume that each vertex is represented by a number, implement a function which reads an edge list represented graph from a file (`graph0.txt`) and stores it as an (i) Edge List and (ii) Adjacency List. Each line of the graph represents an edge from first vertex number to second vertex number in the line.

Exercise 2: Implement a Topological Sorting Algorithm for Directed Acyclic Graphs: i.e. define an interface (in a .h file):

```
List topoSort(Graph G) //returns a sorted list of vertices
```

Write two different implementations (in different .c files) of this function – one using Edge List, and another using Adjacency List as representation.

Exercise 3: Write a main program, with command line arguments for an input filename and an output filename. Your program should read the graph from the input file and write the output list (i.e. sorted list of vertices) into the output file.

Test the algorithm on following three case studies:

i. Synthetic dataset: graph0, 8 vertices, 9 edges
ii. Real datasets
    a. High-energy physics theory citation network
    b. Ubuntu Package dependency graph

Case study 1: High-energy physics theory citation network (`citation.txt`)
https://snap.stanford.edu/data/cit-HepTh.html

Arxiv HEP-TH (high energy physics theory) citation graph is from the e-print arXiv and covers all the citations within a dataset of 27,770 papers with 352,807 edges. If a paper i cites paper j, the graph contains a directed edge from i to j. If a paper cites, or is cited by, a paper outside the dataset, the graph does not contain any information about this. Topological sorting of this graph will help to identify "base papers / most influential papers" of this research area.

Case study 2: GCC dependency graph (`gcc_dependencies.txt`)
https://gcc.gnu.org/git/?p=gcc.git;a=blob_plain;f=Makefile.def;

This example contains a partial Makefile which defines the dependencies between different modules of GCC compiler i.e. this makefile is used to compile gcc source code to install gcc software in a machine. In order to determine the correct order of compilation of modules, the makefile utility does a topological sorting of all targets and dependencies. We have simplified the actual makefile and only presented the dependencies as an edge list. It follows following format to represent that "*moduleA depends on moduleB*":

```
dependencies<space>=<space>{<space>module=<moduleA>;<space>on=
<moduleB>;<space>};
```