CS 340 Programming Assignment III:
Topological Sort

Description: You are to implement the **Depth-First Search** (DFS) based algorithm for (i) testing whether or not the input directed graph G is acyclic (a DAG), and (ii) if G is a DAG, **topologically sorting** the vertices of G and outputting the topologically sorted order.

I/O Specifications: You will prompt the user from the console to select an input graph filename, including the sample file **graphin.txt** as an option. The graph input files must be of the following adjacency list representation where each x_{ij} is the j'th neighbor of vertex i (vertex labels are 1 through n):

```
1:      x11 x12 x13 ...
2:      x21 x22 x23 ...
        .
.
n:      xn1 xn2 xn3 ...
```

Your output will be to the console. You will first output whether or not the graph is acyclic. If the graph is NOT acyclic, then you will output the set of **back edges** you have detected during DFS. Otherwise, if the graph is acyclic, then you will output the vertices in **topologically sorted order**.

Algorithmic specifications:

Your algorithm must use DFS appropriately and run in O(E + V) time on any input graph. You will need to keep track of edge types and finish times so that you can use DFS for detecting cyclicity/acyclicity and topologically sorting if the graph is a DAG. You may implement your graph class as you wish so long as your overall algorithm runs correctly and efficiently.

What to Turn in: You must turn in a single zipped file containing your source code, a Makefile if your language must be compiled, appropriate input and output files, and a README file indicating how to execute your program (especially if not written in C++ or Java). If your program cannot be compiled and/or run on home.cs.siue.edu, you must show me your code running in office hours.

This assignment is due by MIDNIGHT of Thursday, September 28th. Late submissions carry a minus 40% per-day late penalty.