CMP SC 3050 Homework 3
Due: 11:59:59 pm, 03/13/2015

The third homework for CMP SC 3050 is a ***programming assignment***. The first part of the homework specifies the exact problems that your submission should solve and the second part describes the constraints that you **must** follow (no exceptions). Absolutely no late work will be accepted.

Specification:

Your assignment must compile on Babbage in order for you to get any credit.

The assignment should:

- Read in a file representing a directed graph. The first line of the file will represent the total number of vertices, and the remaining lines will be vertex pairs representing edges. We will use strictly positive integers as vertex names, and an edge between two vertices x and y will be represented as (x,y). In the input, edges shall be separated by whitespaces, newlines and tab-spaces. A sample input file can be found on Blackboard.

- Store the directed graph in an adjacency list data structure. Then, write procedure(s) that outputs on stdout, each vertex and the list of vertices adjacent to it in a legible manner. Please use different lines for displaying this information for different vertices.

- Write procedure(s) to check if the direct graph is acyclic. If the graph is acyclic then the display should indicate a topological sort of the graph.

- Write procedure(s) to compute and display all of the strongly connected components of the graph. This display should indicate a list of vertices for each strongly connected component.

Constraints:

The following is a list of constraints for this assignment; failure to adhere to any of these constrains will result in a loss of points or even a zero.

- The assignment should be built using a *makefile*. If you are not familiar with makefiles, contact a TA or come to office hours. The makefile **must** have at least 5 targets: dag, components, ddag, dcomponents and clean.
    1. The target *dag* should, given an input file storing a directed graph, output whether the graph is acyclic. If the graph is acyclic then a topological sort of the graph must be output.
    2. The target *ddag* should given an input file storing a directed graph, output first for each vertex the list of vertices adjacent to it and then output whether the graph is acyclic. If the graph is acyclic then a topological sort of the graph must be output.
    3. The target *components,* should given an input file storing a directed graph, output all the strongly connected components of the graph.
    4. The target *dcomponents,* should given an input file storing a directed graph, output first for each vertex the list of vertices adjacent to it and then output all the strongly connected components of the graph.
    5. The target *clean* should remove all object files and executables from the directory.

- The procedures for the four requirements
    - reading the input,
    - displaying adjacent vertices of all vertices,
    - checking whether a graph is acyclic and computing a topological sort of the graph in case it is acyclic,
    - and computing and displaying the strongly connected components) **must** be written in different files.

- The assignment can be completed in either C or C++. No other programming language is allowed.

- Built-in data structures and external libraries may not be used for this assignment. If the program requires a stack, linked list, or any other structure, it is up to you to provide it. If you have any doubt on whether or not any technique you wish to use is acceptable, do not hesitate to ask.

- A moderate amount of error checking and resource management is required. Your application should ensure that each line from the input file is properly formatted, that the file is successfully opened, the file is successfully closed upon reading of the file and that all allocated space is de-allocated at the exit.

- A moderate amount of formatting and documentation is required. Comments should be descriptive and used to illustrate the purpose and inner workings of an algorithm or function; they should not be used to annotate each line or self-evident logic.

Grading:

There are 23 points possible for this assignment. The grade breakdown is as follows:

- 2 points for correct design of Makefiles.
- 3 points for error checking and resource management.
- 2 points for general programming style and adherence to the constraints.
- 4 points for correctly inputting a directed graph and storing and displaying each vertex's adjacency list
- 4 points for checking correctly whether the graph is acyclic.
- 3 points for outputting a topological sort of a directed acyclic graph.
- 5 points for correctly computing and displaying the strongly connected components of the graph.

If the program fails to compile or crashes due to a runtime exception, a grade of zero will be assigned.