**Project #2 – File Dependency Analysis** due Tuesday, March 06  
version 1.0   
  
Purpose:

This project requires you to analyze package calling dependency structures and store in a package dependency graph. You are expected to use some of the functionality you provided in Project #1. Dependencies between calls have the interesting property that there may be mutual dependency relationships between two or more calls, due to indirect recursion. We call such a mutual dependency set a strong component in the package dependency graph.

A lot of literature has been devoted to efficient algorithms for establishing the existence, and analyzing membership, of strong components. You will be required, as part of this project, to analyze the strong component structure of any graph given in your representation.

Requirements:

Your DEPENDENCY program:

1. **shall** use standard C++[[1]](#footnote-1) and the standard library, compile and link from the command line, using Visual Studio 2010, as provided in the ECS clusters and operate in the environment provided there[[2]](#footnote-2).
2. **shall** use services of the C++ std::iostream library for all input and output to and from the user’s console and C++ operator new and delete for all dynamic memory management.
3. **shall** evaluate package calling dependencies, from a specified file set, based on a type-based analysis[[3]](#footnote-3). File A depends on file B if it:
   1. declares an instance of a type defined in B.
   2. declares a calling parameter or return type defined in B.
   3. inherits from a type defined in B.
   4. calls a global function defined in B.
   5. declares a global variable defined in B.
4. **shall** use a template-based graph facility, meeting the requirements of Project #1, to store package calling dependencies.
5. The graph package **shall** support analysis of strong components[[4]](#footnote-4) for any representable graph, **shall** support the creation of a condensed graph (the graph of strong components), and implement a Topological Sorting algorithm on condensed graphs. These analyses are important for defining test sequences for packages from a large software system.
6. **shall** provide a test executive package and a display package, that, combined with the graph facility, demonstrates you meet all the requirements of this specification. This should demonstrate analyzing a specified file set, finding strong components, and performing a topological sort on the acyclic graph of strong components[[5]](#footnote-5).
7. Your project submission **shall** be uploaded in a zip file archive, including two batch files named compile.bat and run.bat that compile your project and run it using appropriate command line arguments. Please also include a Visual Studio solution that when run demonstrates you meet these requirements.

You should think carefully about the output of this program. The quality of your design is measured, in part, by how well you compose the structure of your output. Note that there is no requirement to provide a graphical user interface. This tool can be implemented very effectively with a command-line input and file and console outputs.

1. This means, for example that you may not use the .Net managed extensions to C++. [↑](#footnote-ref-1)
2. VC++ 2010 is available in all the ECS clusters. [↑](#footnote-ref-2)
3. Types are classes, structs, enums, unions, and typedefs [↑](#footnote-ref-3)
4. <http://www.cs.cmu.edu/afs/cs/academic/class/15451-s06/www/lectures/DFS-strong-components.pdf> [↑](#footnote-ref-4)
5. Your test will be considered weak unless it uses a file set with mutual dependencies. [↑](#footnote-ref-5)