Assignment 11. **Depth first search**

Given April 29, due NEVER (but do it anyway!).

1. The utility `make` works as follows. There is a collection of objects. Each object has an action (e.g. compile, link, run, ...), a list of dependencies, and a time stamp. We say object A depends on object B if B is in the dependency list of A. We say that A depends indirectly on B if there is a chain of dependencies starting at A that includes B. It is not allowed that A depends either directly or indirectly on itself.

a. Show that the collection objects and dependency lists, with the self dependency restriction, forms a DAG.

We "update" an object by performing the action for that object. An object A is "out of date" if there is a B that depends on A so that the time stamp of B is larger than that of A, or if B is itself out of date. For example, suppose A represents an executable program, B represents an object module, and C represents a file containing source code. Suppose the time stamps are respectively 59, 42, and 21. Then these objects are all up to date. Now suppose we edit the source code so that it's time stamp becomes 68. Then B is out of date because an object it depends on has a larger time stamp, and A is out of date because an object it depends on (C) is out of date. We update B by compiling C. This resets the time stamp of B to 73. Then we update A by relinking B. The A time stamp is set to 94. Now all the objects are up to date again; we can run A and get the correct result. Of course, A and B may each depend on several objects.

b. Sketch a DFS algorithm that updates only the objects necessary, as determined by dependency lists and time stamps, for updating a given object, A. Your algorithm should give an error message if there are any illegal circular dependencies. No object should be updated more than once. Assume that during this process, time stamps are not modified except that the time stamp of an object is set to the current time when it is done being updated.

c. Sometimes the action that updates A will modify time stamps of other objects. Make a small modification of your algorithm to determine whether this has rendered A out of date.

2. We have a directed graph, G. We want to write a procedure, `path( vertex u, vertex v)`, that returns the value `BOTH` if there is a path from `u` to `v` and a path from `v` to `u` (that is, `u` and `v` are in the same strongly connected component), `FORWARD` if there is a path from `u` to `v` but not from `v` to `u`, `BACKWARD` if there is a path from `v` to `u` but not from `u` to `v`, and `NEITHER` otherwise. This routine should run in $O(1)$ time. Modify the DFS search algorithm for a directed graph, possibly computing and storing a few extra numbers per vertex, so that this is possible.

3. We have a strongly connected directed graph, G. Modify the DFS search algorithm to remove a many edges as possible subject to the constraint that the graph remain strongly connected. When your procedure is done, removing any further edge would not leave the graph strongly connected.