Project 1 CSc 402 Kutztown

Purpose: Using the STL; Recursion; Graphs & Paths

Due: Date and time TBA, using the turnin script. Late submissions will not be accepted. Setup and test *turnin*. Inability to use *turnin* is not an excuse for late submission.

Description: Finding paths in a graph is a rich topic in our discipline. During class we will look at clever algorithms for finding shortest paths. But, in this project, we are going to get accustomed to advanced data structures by eschewing the efficient in favor of the naïve by designing a brute force recursive solution to the problem of finding all paths from one vertex to another in a directed graph represented by an adjacency matrix of costs.

Start with an STL sequential container to hold a path and place the start vertex in it. Then, passing the container by value, recursively add vertices reachable from the last vertex added until you find a path to the destination vertex (success) or you try to add a vertex already in the path (failure due to cycle). Continue until all possible paths have been discovered.

Each path is to be stored in an ordered (by key) STL associative container where the key is the cost of the path and the value is a container that holds all paths that have that cost. Output occurs after all paths have been found. It is to be ordered by cost, where a cost is printed followed by the vertices of all paths having that cost in a well labeled, cogent manner.

Notes:

- You may use a language other than C++, but you must get approval from the course instructor, and you must use containers available in libraries, c.f. STL for C++, API for Java, etc;.
- Paths may not contain cycles.
- Paths may only occur once in the output.
- A matrix class is provided on acad and the web. All you will use are the >> operator and the *get()* functions. The entire class is provided, however, as this was written by an individual with an interesting coding style that we will likely discuss.
- Be sure the sequential STL container that you use to hold a path maintains the path's ordering.
- Print the paths in ascending order classified by cost. Find an associative container to accomplish this.
- Note: An iterator can't be applied to a const STL object. A const iterator can.
- **GRAD STUDENTS ONLY**: You must also detect when there is no path from the start vertex to the destination. Higher standards will be applied overall, as well.
- This is a senior/graduate level course. Proper style is a must. Substantial penalties, up to and including your program not being graded, will be levied for lazy, incomplete, or chintzy style.
- You must use Doxygen (for C++) or another appropriate documentation tool for your project. Information/tutorials are available on the instructor's links page.

Turnin: The project, to be named paths.cpp (that exact name). Submit a file named **readme.txt** (again, no caps) that includes the link to your Doxygen site. You may also describe design decisions and anything else someone running your program should know.