Project 2 CSc 402 Kutztown

Purpose: Creating Data Structures Using the STL; Graphs & Paths; Analysis
Points: Program: 25 points
Due: Program: 12 Noon on the specified day, using the turnin script. Submit late at your own risk.
Description: Kruskal's algorithm for finding a minimum cost spanning tree is a neat algorithm that employs several data structures on which we should get some practice.

Implement the algorithm using a graph stored in the same format as was done in Project 1. Choose appropriate data structures from the libraries available with your chosen language, e.g. If you use C++ you must use STL objects to the greatest possible extent.

Output the spanning tree (in a manner that makes it readable and understandable) and the cost. The tree should include the edges included and the cost of each. The output must be well formatted and easily interpreted.

GRADUATE STUDENTS <u>ONLY</u>

Implement the Kruskal algorithm a second time using as few library data structures as is reasonable. You should use a binary heap for the priority queue (see note below). You need not implement any ADTs from scratch.

Refine your programs to output the actual clock time the Kruskal algorithm required, in microseconds. Then, run real-world tests to see how well your program performs. Create test graphs that have potential to behave differently when used with different languages and the two versions you are writing. Also, update your program to write times to a file.

Use cExec (explained in class) to run several tests of 10 runs each of each of your programs with your selected graphs to permit apples to apples comparisons.. Provide the raw data on one tab and present the data on another tab. Your readme should have a detailed analysis that draws conclusions regarding whether your priority queue or other data structure selections appeared to have an effect on timings, if there is an appreciable difference. It should describe the data, why you chose that data, the results, and conclusions. It should be at a level appropriate for a graduate student, i.e. don't be lame, explain.

Notes:

- Be sure (read the STL or similar documentation) that your priority queue implementation and operations (the STL has specific heap operations in Section 5.3) is *log n*.
- The program should be runnable in batch mode if a file name appears on the command line.
- Be sure you have properly read the file before trying to process it.
- An STL *priority_queue* object can be constructed with a comparator. You may need this to assure proper handling of edges and costs in the queue.
- A graph can have cycles. This should be inconsequential.
- You must detect an unconnected graph.
- Discussions of test sets to best evaluate the programs should occur in the forum among all students. Forum participation is part of your grade.
- GRADS:
  - You may use the heap example in my 237 area that is implemented with a vector.
- To get microsecond precision in C++ requires using C++ 11; our default version of g++ incorporates it. There is an example named *Microseconds* in the instructor's CSc402/Examples/Microseconds directory on the web and acad. It requires use of the *chrono* library.

Deliverables:

D2L: Submit a file named **readme** (pdf, docx, etc.) that includes the link to your Doxygen website in the project dropbox. You may also describe design decisions and anything else someone running your program should know. Grads must include their analysis.

turnin: Named kruskal.cpp (that exact name if in C++). Provide a makefile and directions to run it, if appropriate. The executable is to be named *kruskal*.