COLORADO MESA UNIVERSITY

CS480 Assignment 2
Due Date:October 1 - 11:00PM

This assignment covers the following topics:

- Graphs
- MST Algorithms
- Shortest path

Remember that:

1. **The program must compile error and warning free (do not suppress warnings) and run on Linux using a makefile, only the GCC, G++ or Clang++ compiler allowed.**
2. **Programs that are not able to be compiled will not be considered for a grade.**
3. **main.cpp source code file header should contain the academic integrity statement.**
4. **Functions should be preceded by a comment block outlining parameters return values and a brief algorithm description.**

Task 1:

Decide whether these statements are True or False. 10%
You must justify your answers to receive full credit.

(a) If some edge weights are negative, the shortest paths from s can be obtained by adding a constant C to every edge weight, large enough to make all edge weights nonnegative, and running Dijkstra's algorithm.

(b) Let ***P*** be a shortest path from some vertex ***s*** to some other vertex ***t***. If the weight of each edge in the graph is squared, ***P*** remains a shortest path from ***s*** to *t*.

Task 2: Problem Specification: 90%

For the second task you are required to write a program "MST.cpp" will take as input a map rendered as graph data. The map is of the major and capital cities of post WWII USA and Canada (June 1949). The vertices are points in the plane and are connected by edges whose weights are Euclidean distances between the points. Think of the vertices as cities and the edges as roads connected to them.
In the "graph", each pair of cities is connected by an undirected edge representing the distances between the cities. The minimum spanning tree for this graph will represent 127 roads between cities that allow all cities to be connected to each other. There are many such sets of 127 roads (spanning trees); you are to find the set that uses the minimum asphalt (hence, minimum spanning tree).

The complete data set is in distances.txt. The format of the data file is as follows:

- The first 2 lines are comments (to be ignored).
- Each of the 128 cities is represented as:
    - The name of the city (state), on a separate line. Ignore everything after the opening brace (latitude, longitude, and population).
    - A series of zero or more lines that contains the distance, to each of the previous cities in the list, in the reverse order of those cities.
    - The last line is a comment.

Your program should use Kruskal's algorithm to determine the minimum spanning tree. You should consider using a PriorityQueue (you can use the Linked List you developed in weeks 1 and 2) to allow selection of edges by order of weight. A heap may also be useful but I leave the details up to you.

Your output should give the total cost of the minimum spanning tree and then list the (127) edges in the minimum spanning tree in the format: City (State) to City (State) "distance = ", one line per pair.

In writing your code, you may assume that the data file is formatted as described above, but you may not assume a limit of 128 cities. The number of cities will not be known until you have finished reading the file. **You may only read the file once.** So, I would expect that you will need to use some form of List rather than arrays or matrices.

The following command should render a minimum spanning tree of graph described in the data file:
$./MST distances.txt "start city" and
$./MST distances.txt "start city">output2.txt should produce a text file with your program output clearly laid out for evaluation.

Test your program with these command in a Linux environment (this must work).

Your programs should output the following:

1. Compile via a makefile to 1 of 2 possible targets **10%**
2. **Target 1** - "MST" will accept 1 argument being the "distances.txt" file, and start city and produce MST as described above
    a. The minimum spanning tree for 128 cities or a subset thereof. **25%**
    b. Determine the time complexity of your solution **5%**

3. **Target 2** - "journey" will accept 3 arguments the first "distances.txt" the 2nd and 3rd arguments being a start city and a destination city, it will produce a leg by leg description of the shortest path between the 2 cities for use by a long hauler (intercity truck) driver.
    - The shortest path and related data for the long hauler should be in the format 1 leg per line with line numbers (ordered start to finish). **35%**
    - Determine the time complexity of your solution (show workings). **5%**

Code your algorithms as efficient as possible. Use appropriate comments in your source code.

I. A note on object orientation and the STL and specifications

You may use classes/modules for your code if you wish, but be careful that splitting the program into objects does not interfere with the readability of the algorithms. If I find that your algorithms are difficult to follow (due to being spread across multiple parts of the code or for any other reason), your demo time will be rather extended. **5%**

Blacklisted: Any/all STL libraries for this assignment. **-25%**

- The use of strings and all related functionalities within the string library is permitted. All data structures and related algorithms should be coded yourself.
- A 25% penalty will be applied for failure to comply with specifications such as filenames, makefile directives, unexpected interactivity, outputs etc..
- A non-compile or fail to run in a Linux environment shall adversely effect your outcome.

- Presentation, coding style detailed comments expected **5%**

II. Submissions

- You should submit both your source code (.cpp, .h) and a **Makefile** named "makefile" that makes both your programs using Clang++, CC or g++. The target will be specified at the compilation time by the user eg: $ make MST, $ make journey, make clean or $make all.
- The resulting programs should be named "MST", and/or "journey" (case sensitive).
- The command "./journey distances.txt <departure city> <destination city>" in a Linux terminal should run your program end to end with no user interaction of any kind.
- The command "./journey distances.txt <departure city> <destination city> >output.txt" should result in a text file named output.txt containing the reports resulting from your program run, please ensure the output is properly laid out for ease of evaluation.
- The command "./MST distances.txt > <start city> >output.txt should result in a text file named output.txt containing the MST properly laid out for ease of evaluation.
- Submit using the D2L dropbox 1 only archive TGZ (compressed tarball - something like this would do it: $ tar -czvf Assignment1.tar.gz /home/yourUID/assignment1/)
- No email submission can be considered- sorry!