**Problem:** In this assignment, you will implement an algorithm to find the shortest path between two vertices of an undirected graph whose edge weights satisfy the triangle inequality.

**Running the program:** Your program will be run as follows:

dijkstra <input-graph-name> <u> <w>

**Input file:** The input files will have the following format, and the output should be the length of the shortest path from vertex <u> to vertex <w>, followed by the actual path, sent to standard out.

| **Input graph** | **Output** |
| --- | --- |
| n | p |
| v1 s11 w11 s12 w12... s1k1 w1k1 | t1 t2 ... tk |
| . |  |
| . |  |
| vn sn1 wn1 sn2 wn1 ... snkn wnkn |  |

Here, *n* is the number of vertices, *vi* are strings containing the names of vertices, *sij* is a string giving the name of the *j*th neighbor of vertex *i* in the graph, *wij* is a float number giving the weight of the corresponding edge, *p* is the length of the shortest path, and *ti* is the *i*th vertex on a shortest path from <u> to <w>.

**Example:** *dijkstra Test-file f cd*

| **Test-file** | **Sample output** |
| --- | --- |
| 4 | 5.1 |
| ab cd 4.2 e 1.0 f 1.0 | f ab e cd |
| cd ab 4.2 e 3.1 |  |
| e ab 1.0 cd 3.1 f 2.3 |  |
| f ab 1.0 e 2.3 |  |

**Files:** You should create a directory called hw4 on linprog. This directory should contain the following files.

1. *dijkstra.cpp*: Contains the implementation of the main program. You may include additional files for your code.
2. *Makefile*: The makefile for this assignment. Typing *make* should create the executables: *dijkstra*.
3. *Discussion.pdf*: Please run your code on a graph that arises from some real world application, using real world data, and report the results you get, in one or two pages, in this document. Your report you should mention the following. 1. The application you considered, identifying what the vertices, edges, and edge weights represent (for example, vertices are cities, edges are roads between cities, and weights are driving distances). 2. The size of the graph in terms of the number of edges and number of vertices. 3. The source of your data. 4. Any optimizations that you performed and the performance improvements due to them. 5. Summary of your findings.

**Submission procedure:**

1. You should maintain a development log for your program. This will mention a list of features you implemented, bugs you fixed, and other significant changes you made, each day that you worked on the code. If you received help from others (you are allowed to get a small amount of help from others for debugging your code, though you may not copy any code), please list that too in the development log. This will be a plain ASCII text file, called LOG.txt.

Here is an example:

Development log: Assignment 4

Nov 12, 2021: Wrote header files for the classes: LinkedList, PlayGame, ChessBoard, Player, and Timer.

Nov 13, 2020: Implemented the class PlayGame, and tested it.

Nov 14, 2021: Implemented insert and delete functions in LinkedList.

Nov 15, 2021: Implemented the IsMember and IsEmpty functions in LinkedList. Tested the LinkedList class.

Nov 16, 2021: Implemented the ChessBoard and Player classes, and tested it.

Nov 17, 2021: Changed the implementation of LinkedList to use a doubly linked list.

Nov 18, 2021: Implemented the Timer class, and the main function. main did not compile -- linker said the class LinkedList was undefined.

Nov 19, 2021: Deva Datta helped me fix the LinkedList link error. It was because it was defined in a different namespace. Code compiled but seg-faulted.

Nov 20, 2021: Fixed the seg-fault by tracing execution using ddd. I was trying to dereference an invalid pointer. This in turn was caused by an uninitialized variable. Tested the code on many test case I created. Appears to run correctly.

Nov 21, 2021: Performed more tests, and submitted the assignment.

1. Make sure that your code compiles and runs correctly. We will test your code on linprog (a Linux PC), using the g++ compiler. Your code should work correctly on this machine, and so you may want to develop your code on it.
2. You should create a directory called projX\_fsuid, where X is the project number and "fsuid" is replace by your FSU ID. For example, if your FSU ID were kb14ac, you would create a directory called proj4\_kb14ac for programming assignment 4. Copy all required files for your assignment (typically a Makefile, a LOG.txt file, and .cpp and .h files) to this directory. Do not copy executables.
3. Create a tar or zip file for this directory. Then submit the tar or zip file on Canvas.
4. Do not make any modifications to your files after the deadline until the assignment is graded. That way, we will be able to tell from the timestamp on the files that you had actually completed your assignment on time, in case there is any problem with your assignment submission.

**Grading:** You will be graded on the following (i) correctness of your code, (ii) performance of your code, and (iii) the quality of your report (presentation and content). You will get 10 bonus points on this assignment if your code is the fastest in class on test cases that we run. You may also get bonus points if you perform interesting optimizations.