COMP 3761 Assignment 7

Due: Wednesday **March 18, 2015** at 6:30pm

1. Textbook Page 322 Exercise 9.1 #3. [5 points]
2. Textbook Page 323 Exercise 9.1 #4. [5 points]
3. Textbook Page 323 Exercise 9.1 #5. [5 points]
4. In this assignment, you'll implement Dijkstra's shortest-path algorithm. [10 points]

   - Download the text input file **DijkstraData.txt**.

     The file contains an adjacency list representation of an undirected weighted graph with 200 vertices labeled 1 to 200. Each row consists of the node tuples that are adjacent to that particular vertex along with the length of that edge.

     For example, the 6th row has 6 as the first entry indicating that this row corresponds to the vertex labeled 6. The next entry of this row "141,8200" indicates that there is an edge between vertex 6 and vertex 141 that has length 8200. The rest of the pairs of this row indicate the other vertices adjacent to vertex 6 and the lengths of the corresponding edges.
   - Your task is to implement and run Dijkstra's shortest-path algorithm on this graph, using 1 (the first vertex) as the source vertex, and to compute the shortest-path distances between 1 and every other vertex of the graph. If there is no path between a vertex v and vertex 1, we'll define the shortest-path distance between 1 and v to be 1000000.
   - Program Output:

     You should report the shortest-path distances to the following ten vertices ONLY, in order: 7, 37, 59, 82, 99, 115, 133, 165, 188, 197.
   - IMPLEMENTATION NOTES:

     This graph is small enough that the straightforward $O(mn)$ time implementation of Dijkstra's algorithm should work fine. However, you are required to implement the heap-based version (using the priority queue) that runs in $O(m \log n)$ time-efficiency, where n is the number of verticies, and m is the number of edges in the graph.

     Note this requires a min-heap that supports deletions, and you'll probably need to maintain some kind of mapping between vertices and their positions in the heap.