CSCE 221 Assignment 6 Cover Page

First Name Last Name UIN

User Name E-mail address

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)				
Printed material				
Other Sources				

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Signature: _____

Your Name Date

CSCE 221 Final Project (Assignment 6)

Replacement for the final exam

Due December 01, 2013

• Problem Description

Use the Dijkstra's algorithm to find a shortest path in a directed graph from a vertex called "source" to all other vertices. The solution should provide a value of the shortest path along with all intermediate vertices and weights on the path.

Remark: The Dijkstra's algorithm provides a solution to the problem using a graph with positive weights.

• The project requirements

- Implement a graph data structure using a collection of adjacency lists such as it is represented in the textbook, see the Figure 13.4, p. 603. You may use the C++ standard library classes, `std::vector` or `std::list`, when you work with arrays or linked lists, respectively. At least the following graph operations should be included:
  - inserting a new edge or a vertex: `addEdge` and `addVertex`.
  - getting a list of all vertices.
  - finding a vertex adjacent to a given one.
- Implement a minimum priority queue represented as a minimum binary heap; reuse your code from the programming assignment #5. You can choose one of these options:
  - a minimum binary heap without locator (90% of the maximum score)
  - a minimum binary heap with locator (100% of the maximum score)
- Implement the Dijkstra's algorithm using both the data structures: graph and minimum priority queue.

• The input output requirements

- Create a graph to be loaded from a user-defined data file. Use the following file format. (Here field descriptions are in comments but you do not need to use comments in your data file.)

```
 3   /* = the number of vertices */
     /* start numerating vertices from 1 */
     /* for a given vertex list all adjacent vertices and weights end by -1 */
     /* format: vertex, adjacent_vertex, weight, adjacent_vertex, weight,..., -1 */
 1  2  5  3  2 -1
 2  3  4  -1
 3  -1
-1         /* end of file */
```

- Display the graph in a similar format on the screen.
- Ask the user about the source and destination vertices in a graph.
- Display the value of the shortest path along with all the intermediate vertices with weights starting at the source and ending at the destination. Example:

```
      2       6       1
S ----> A ----> B ----> D
Total wight of the shortest path from S to D = 9
```

- Display the total number of comparisons in order to compare the performance of the Dijkstra's algorithm. You should count
  - the number of comparisons to remove minimum and restructure the heap after this operation
  - the number of comparisons used to find/update and decrease element key value in the priority queue and its heap representation.

- **Report**

  In addition to the regular programming assignment report you should include answers to the following problems.

  1. Write about the running time of the Dijkstra's algorithm related to your implementation of Graph and Priority-Queue ADT. Use the big-O notation to express the running time in order to compare performance of the implementation of the Dijkstra's algorithm with the most efficient complexity of this algorithm.
  2. Discuss the implementation of the Dijkstra's algorithm related to the representation of the Priority Queue ADT with and without locator. What is the impact of implementing the Priority Queue with locator on the complexity of the Dijkstra's algorithm? Use the big-O notation to express the running time in each case.
  3. Write about how you used the Priority Queue ADT to simplify coding and obtaining the most efficient implementation of the Dijkstra's algorithm if it applies to your implementation.
  4. Test the Dijkstra's algorithm on different graphs. In your report, find a solution for each graph by hand and provide computational results.
  5. Write about three real-life applications where you can use the Dijkstra's algorithm.