CSCE 221 Cover Page

Programming Assignment #6

**Due: 12/11/2019 11:59pm**

**Important Notes:**

1. **This is an optional project to give you the opportunity to secure a best 5 out of 6 project score on the overall project grade. If you are satisfied with your project performance in PA1-PA5, , you do NOT have to implement this project.**
2. **There is one hard deadline of Dec 11 for this project (i.e. the late submission policy does NOT apply). No extensions will be allowed.**

First Name: Vidhur Last Name: Potluri UIN: 626007235

**Any assignment turned in without a fully completed cover page will receive ZERO POINTS.**

Please list all below all sources (people, books, webpages, etc) consulted regarding this assignment:

CSCE 221 Students Other People Printed Material Web Material (URL) Other

1. 1. 1. Course Materials 1. 1.

2. 2. 2. 2. 2.

3. 3. 3. 3. 3.

4. 4. 4. 4. 4.

5. 5. 5. 5. 5.

Recall that University Regulations, Section 42, define scholastic dishonesty to include acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion. Please consult the Aggie Honor System Office for additional information regarding academic misconduct – it is your responsibility to understand what constitutes academic misconduct and to ensure that you do not commit it.

I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received nor given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment.

Today’s Date: 12/7/2019

Printed Name (in lieu of a signature): Vidhur Potluri

**Graphs**

**Description**:

For this programming assignment, you will implement a Graph using an adjacency list representation and then perform Dijkstra’s shortest path algorithm on the graph.

Your program will read a graph from an input .txt file that describes graph connectivity. The file is a text file where the first line contains two numbers. The first is the number of vertices *n* and the second is the number of edges *m*. After this line there will be *m* lines with three numbers. The first two numbers represent the source and destination vertex for the **undirected** edge. The third number is the weight for that edge. The final line of the file contains two numbers representing the index of two numbers (start and end vertices in the shortest path calculation)

Your program should construct the graph and run **Dijkstra’s shortest path algorithm**.

An example input file:

4 5

0 2 1

1 2 5

2 3 3

1 3 2

0 3 10

0 3

This file represents a graph with 4 vertices, 5 edges, and has edges (0,2) with weight 1, (1,2) with weight 5, (2,3) with weight 3, (1,3) with weight 2, and (0,3) with weight 10. **Your program should output the shortest path between vertex 0 and vertex 3 as a sequence of vertex labels 0, 2, 3 (in this example).**

**NOTE**: The weight does NOT have to be an integer. In general the weight will be a floating point number. *An example graph and its solution is provided as a reference for you to test your code.*

To implement the graph please start with the included implementation **Graph.h** filling in the specified functions. Your program should read the graph in the input txt file, and output the solution in the format from the example above. Your shortest path algorithm will also need a **Heap using locators (included with this assignment)**. As you insert items into the heap, you will need to store the locator for each vertex in the vertex itself.

**Coding Portion** **(100 Points):**

* Start with the following template: **Graph.h** and fill in the entire member functions or implement your own version of the graph.
* You should implement the adjacency list data structure for the graph.
* Be sure to test the correctness of your algorithms and implementations.
* Your code will be graded based on whether or not it compiles, runs, produces the expected output, produces correct output, and your coding style (does the code follow proper indentation/style and comments).
* Please be sure to turn in the cover page.