CS 260
Programming Assignment 9

This lab continues the practice with graph traversals, but considers weighted graphs.

The equivalent of a minimum spanning tree for a weighted graph is a min-cost spanning tree. This is a tree with the minimum total cost to reach all nodes. You can do this either with a directed or non-directed graph, for this exercise, you will use a non-directed graph.

For this exercise, you need to modify the graph class used in your previous lab to support edge weights. The addEdge method will need a third argument, an integer that shows the weight of that edge. You will also need to modify the edge structure, the connectivity table, and the related display methods.

As described in Moodle, a min-cost spanning tree is created with a breadth first search that uses a priority queue of edges instead of a FIFO of nodes. When you add a new edge to the priority queue, check the queue for another edge with the same ending node and only leave the edge with the lowest cost. Since you are going to be searching the priority queue, you should implement your own based off of an array rather than using a heap. (Note that the edge structure will also need to have beginning and ending nodes, rather than only ending as in the provided Graph class.)

Remove the lowest cost edge in the priority queue, add the edge to your tree, mark the ending node as visited, and add any edges from it to unvisited nodes to the priority queue.

Your output should show the resulting edge list like this: A-B B-C A-D D-E C-F B-G
To work with the driver, you will need to name your class WGraph, it will need the same methods as in the Graph class of Lab 8 with the addition of the method:

```
std::string minCostTree();
```

This work by Jim Bailey is licensed under a Creative Commons Attribution 4.0 International License.