CSE4102: Programming Languages **Spring 2017**

Homework 6: C Programming

Out Date: 03/23/2017
Due Date: 03/30/2017

Objectives

This is an adaptation of Problem 5 from the Northeast Regional Preliminary Competition of the 2014 ACM ICPC.

Airlines face a difficult problem of ensuring that their passengers can get between each of the airports they serve, in some number of steps, while simultaneously trying to minimize their costs.

A new airline, CSE4102 Airways, has already selected the set of airports they plan to serve. However, they have two tasks remaining:

- Given the costs of all routes they could possibly serve, figure out the cheapest routes that allows any passenger to get from one airport to another in some finite number of steps.
- Select their hub airport. The hub should be the airport which requires the fewest number of combined flights to reach every other airport.

For example, the table below shoes the set of flights available, and their costs. The starred flights indicate the ones that CSE4102 Airways would choose to operate. The starred airport code is the one they selected as their hub. Note that the table below (and the file input) omits the symmetric entries. Namely, while the PVD row has no entries, the PVD column specifies connectivity with 2 airports $\{JFK, ORD\}$ which implies that you can take those two flights in any direction.

AP	BOS	BWI*	DFW	JFK	LAX	MIA	ORD	PVD	SFO
BOS	-	-	-	187*	-	1258	867	-	2704
BWI*	-	-	-	184*	-	946*	621*	-	-
DFW	-	-	-	1391	1235*	1121	802*	-	1464
JFK	-	-	-	-	-	1090	740	144*	-
LAX	-	-	-	-	-	2342	-	-	337*
MIA	-	-	-	-	-	-	-	-	-
ORD	-	-	-	-	-	-	-	849	1846
PVD	-	-	-	-	-	-	-	-	-
SFO	-	-	-	-	-	-	-	-	-

With this arrangement of flights, BWI is selected as the hub. From BWI, three airports are reachable in 1 step (ORD, JFK, MIA), three are reachable in 2 steps (PVD, BOS, DFW), and one in 3 steps (LAX), and one in 4 steps (SFO). The sum is 1+1+1+2+2+2+3+4=16, which is the minimum in this arrangement.

Because the point of this class is to expose you to various languages and not test your algorithmic prowess we will be walking you through the steps of creating a solution to this problem.

For all three questions, we are providing header files and your task is limited to implementing the abstract data type. In particular, you do receive the content of `main.c` which implements the overall program. You are also receiving a `Makefile` to build the project. It is completely straightforward to build this code on Linux or MacOS. On Microsoft Windows, it may prove more challenging to get the tools. We *suggest* instead to use a virtual machine and do the work under Linux. Naturally, if you are already quite comfortable with Windows tools, it should be easy to use those.

When you are done, zip the directory and submit the zip file as your solution.

Question 1: Graph ADT

The first task is to implement the required functions prototyped in graph.h for reading in the data from a test file and constructing a valid representation of the undirected graph.

Test cases are formatted as such: the first line contains a single integer, denoting the number of airports in the graph, followed by a newline character. The remaining lines will be composed of two three letter airport codes separated by a single whitespace followed by an integer and a new line character. Each line represents a bidirectional flight containing two airports whose flight cost is given by the numeric value. Consider the sample input below as an example.

```
9
ORD BWI 621
ORD BOS 867
BOS SFO 2704
ORD SFO 1846
BWI JFK 184
MIA BWI 946
JFK MIA 1090
BOS JFK 187
ORD DFW 802
LAX MIA 2342
PVD JFK 144
MIA BOS 1258
LAX SFO 337
DFW JFK 1391
DFW SFO 1464
ORD JFK 740
DFW LAX 1235
MIA DFW 1121
PVD ORD 849
```

Note that the resulting graph would be

```
ORD : BWI BOS SFO DFW JFK PVD
BWI : ORD JFK MIA
BOS : ORD SFO JFK MIA
SFO : BOS ORD LAX DFW
JFK : BWI MIA BOS PVD DFW ORD
MIA : BWI JFK LAX BOS DFW
DFW : ORD JFK SFO LAX MIA
LAX : MIA SFO DFW
PVD : JFK ORD
```

Question 2: Priority Queue ADT

Implement functions for creating a priority queue using a heap representation to ensure a runtime for the `extractMin` of $\Theta(\log n)$ where n is the number of items in the priority queue.

Question 3: MST

Implement a minimum spanning tree algorithm to construct (in a new graph) the MST for the initial graph. Clearly your MST construction will use the priority queue you created earlier.

The motivation to compute an MST is quite straightforward. An MST gives all the cheapest connections spanning all the vertices of the graph. If you are to select a distinguish vertex as the "root" of the MST, you would see how many *hops* are needed to reach any other vertex in the graph starting from that root. Indeed, you would locate all the airports 1-hop away and from those the airports 2 hops away, etc...

On the example described earlier, the MST would be

```
JFK : PVD  BWI  BOS
PVD : JFK
BWI : JFK  ORD  MIA
BOS : JFK
SFO : LAX
LAX : SFO  DFW
ORD : BWI  DFW
DFW : ORD  LAX
MIA : BWI
```

Question 3: Optimal Hub Selection

To compute the optimal hub, you can simply try every single vertex as the root of the MST and evaluate how many hops are needed if that vertex was the hub. Then you simply return the best vertex (fewest hops).

As you can see in the example, designating a specific airport as the hub yield a different "orientation" of the MST and a different number of hops to reach all vertices. The output of the program should simply be the name of the selected hub.

Have fun!