COT5405: Programming Assignment 1 (Spring 2016)

Assigned: Mar. 1, 2016

Due: Mar. 18, 2016

Instructions:

For all programming assignments, you shall write program code in one of the following languages: C, C++, Java, or Python (those that can be executed on the Eustis2 Unix machines). Note: it is your responsibility to know how to log in to the Unix machine in our Computer Science Division, how to code and run your programs on this machine.

The graph below will be used for all the following questions.

Input:

Your programs should be designed to read input from data files (text files). For these programs, the input is the adjacency-list representation of the graph (one of the results of Question 1). Your programs will be tested with, in addition to the input given below, data designed to see how your programs handle exceptions and bad input data. A program run on 'good' input should generate correct, properly formatted output.

Output/Results:

Your SSH client window should display on screen execution results at the end of the program run.

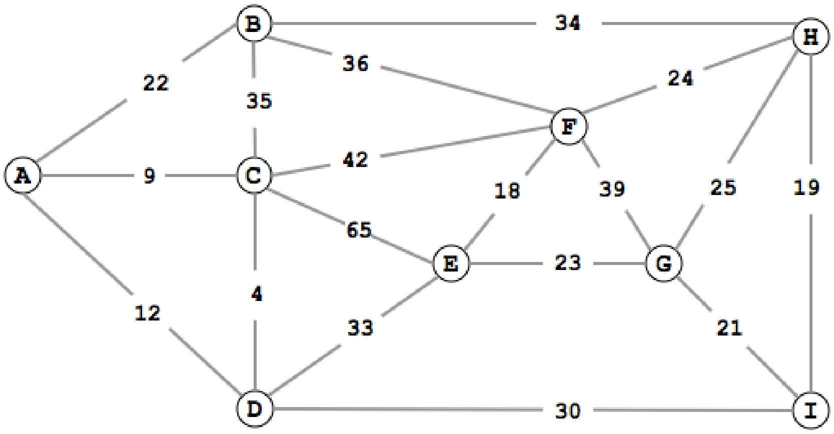
The ***screenshot image*** of your SSH client window containing the result printout should be captured and presented in the result section of your report.

Submission:

1. A project report containing the following:
    a. The purpose of the assignment and a brief introduction of the problems
    b. Method: language used, compilation instructions, format of input, platform specifics (i.e., Unix), algorithms to solve the problems, etc.
    c. Outputs of test input data
    d. Discussions/Remarks on running times, implementation problems, possible implementation improvements
2. A folder/directory for each of the three programs in Question 2 to 4 is created. Self-contained: The source code of each (along with any auxiliary files, e.g., input data) should be contained in its folder so that the code can be compiled and run from the folder.
3. Compressed all files and directories into one zip/rar file and submit it online by the due date.

Problem Description:

Consider the following undirected graph with weight values on each edge.



1. **(No programming question**) Write down the adjacency matrix representation and link list representation of this graph (in the format similar to the examples of Pages 8 and 9 on lecture slides 'graph.ppt'). The edge value should be shown in those two representation formats, too. Use the adjacency-list (i.e., link list) presentation of the graph as the test input to your programs.

2. **(Heap and Heapsort**): Considering all the edge values in above graph as an unsorted list. Design a program to heapsort this list of values in ascending order. Use the array data structure for the heap (as shown in Page 20 in lecture slides 'heapsort.ppt'). Your code should print out this array after the initial "heapify the array" operation (as the array example on Page 20 of slides), and the final sorted result list. In addition, in your report, you should draw the heap binary tree (as the tree example on Page 20 of slides) according to the initial heapify result.

3. **(Shortest Path**): Write a program implementing Dijkstra's Algorithm to generate the shortest path results starting from node A to all other nodes. Your code should print out the shortest path route and the path value for each destination node. For example, if the shortest path from node A to node E is A -> C -> E, then your code should print out result for this destination node as:

   ```
   Destination Node E:  path value = 74, path is: A -> C -> E
   ```

4. **(Minimum Spanning Tree**): Implement Kruskal's Algorithm to generate one MST for this graph. Suppose the MST contains three nodes (A,C,E) and the two edges between them, then your code should print out result like:

   ```
   Minimum Spanning Tree: Total weights on MST edges = 74
   Node Set = {A, C, E}, Edge Set = { A-C,  C-E}
   ```