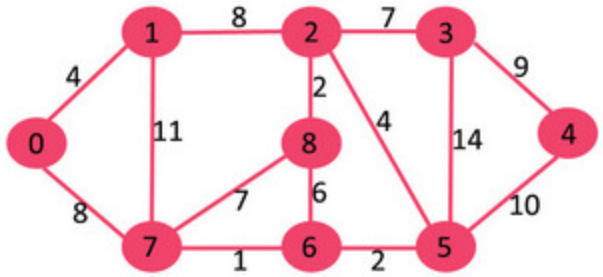
CSC 3430 Algorithm Design and Analysis
Homework 4: Implement Dijkstra Shortest Path Greedy Algorithm
Due: Friday, February 17, 2017

Design and implement a C/C++ solution of the Dijkstra Shortest Path greedy algorithm for an undirected weighted connected graph with no self-cycles. Your solution must determine the shortest-path distance and the specific sequence of vertices that make up the shortest-path from any starting vertex in a graph to each of the other destination vertices.

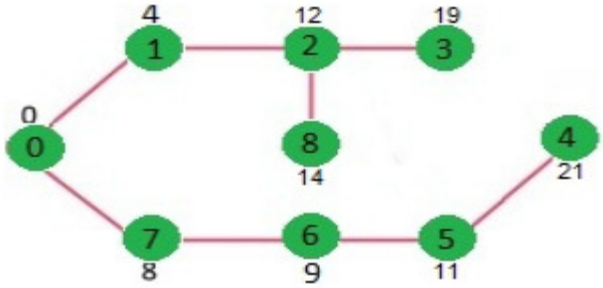
For example in the following graph:




```
// Number of vertices in the graph
#define V 9

int graph[V][V] = {{ 0,  4,  0,  0,  0,  0,  0,  8,  0},
                   { 4,  0,  8,  0,  0,  0,  0, 11,  0},
                   { 0,  8,  0,  7,  0,  4,  0,  0,  2},
                   { 0,  0,  7,  0,  9, 14,  0,  0,  0},
                   { 0,  0,  0,  9,  0, 10,  0,  0,  0},
                   { 0,  0,  4, 14, 10,  0,  2,  0,  0},
                   { 0,  0,  0,  0,  0,  2,  0,  1,  6},
                   { 8, 11,  0,  0,  0,  0,  1,  0,  7},
                   { 0,  0,  2,  0,  0,  0,  6,  7,  0}
                  };
```

the following is a visual representation of the shortest-path distance to each destination vertex starting from vertex 0:



and your program output starting from vertex 0 might look like:

```
Vertex   Distance from Source: 0
0                0       Path: 0
1                4       Path: 0-->1
2                12      Path: 0-->1-->2
3                19      Path: 0-->1-->2-->3
4                21      Path: 0-->7-->6-->5-->4
5                11      Path: 0-->7-->6-->5
6                9       Path: 0-->7-->6
7                8       Path: 0-->7
8                14      Path: 0-->1-->2-->8
```

or starting from vertex 8 instead:

```
Vertex   Distance from Source: 8
0                14      Path: 8-->2-->1-->0
1                10      Path: 8-->2-->1
2                2       Path: 8-->2
3                9       Path: 8-->2-->3
4                16      Path: 8-->2-->5-->4
5                6       Path: 8-->2-->5
6                6       Path: 8-->6
7                7       Path: 8-->7
8                0       Path: 8
```

Some details and requirements:

- Your program must include a representation of a Graph data structure.
  You can decide whether or not to use a C++ Class to encapsulate this, or to just implement appropriate array / struct data structures at the main() program level.

  The adjacency-matrix representation is recommended, although the adjacency-list representation is also an acceptable alternative for this lab.

- After creating and initializing the Graph data structures, your main( ) program must then run the Dijkstra Shortest Path greedy algorithm on this graph several times in a row, each time beginning with a different vertex as the "starting" vertex.

  Note: if the Graph has K vertices, then running your lab will cause the analysis and output all K different shortest-path analysis solutions for that graph.

  As a suggestion, your code should include a function/method such as
  dijkstra(Graph, startingVertexNumber)

  Each time it is called, it computes the minimum distance and shortest-path sequence of vertices from the startingVertexNumber to each of the other vertices in the graph, and outputs a summary report for that starting vertex similar to the ones shown in the above examples.

- Use the specific graph shown above (the top graph showing the vertices and edges/weights) as the data for this lab. Initialize your graph data structure with the vertex/edge/weight information from the diagram. You do not need to implement the logic to read input data from the user related to this graph structure; instead you can simply hard-code the initialization information for this particular graph example into your code. This must, of course, be easy to modify to adapt your program to a different graph with a different number of vertices and edges.

- It is recommended that you study and refer to the following online articles as you begin work on this lab. You may use any of the ideas or code detail in these articles that you like, although this is not a requirement for this lab assignment. If you do use specific code details, be sure to include appropriate acknowledgements in your code. Note that some of the specific details in the example code may have errors – check carefully before using.
  http://www.geeksforgeeks.org/tag/Greedy-Algorithm/

  http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/

  http://www.geeksforgeeks.org/greedy-algorithms-set-7-dijkstras-algorithm-for-adjacency-list-representation/

Turn in on due date:

1) Source code listings for all of your solution code. Print code using fixed-pitch font (e.g., Courier New), and landscape mode for the layout orientation.

2) Execution run output showing the K different shortest-path analysis solutions for the specified graph that is used as data for this lab.