CSC 323 Algorithm Design and Analysis, Spring 2018
Instructor: Dr. Natarajan Meghanathan
Project 8: Use the Results of Breadth First Search to Extract the Shortest Paths from a Particular Vertex to the Rest of the Vertices in an Undirected Graph

Due: April 19, 2018: by 1 PM (in Canvas)

The objective of this project is to use the Breadth First Search (BFS) algorithm to determine the shortest paths (and print them) from a particular vertex to the rest of the vertices in the graph. You are given the code for running the BFS algorithm starting from a particular vertex (say, vertex 0). The Graph class has member variables to keep track of the predecessor vertex for every vertex (in the form of the predecessorList array) on the BFS tree rooted at the starting vertex. The code given for the BFS algorithm also updates the entries in the predecessorList array as part of the traversal.

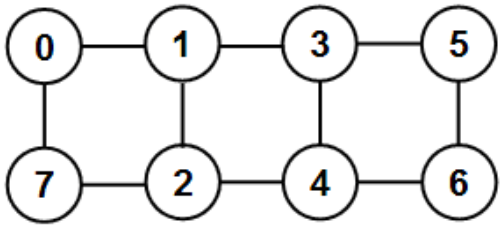
Your task in this project would be to use the results (like the entries in the predecessorList array) of running the BFS algorithm starting from vertex 0 and write an iterative or recursive function that would print the actual shortest paths (sequence of edges) from vertex 0 to the rest of the vertices in the graph. You could add the iterative or recursive function as part of the Graph class and access it from the main function. Your main function can also print the predecessor for every vertex in the BFS tree rooted at vertex 0, as shown in the sample output.

A sample output (printing the predecessor entries and the shortest paths from vertex 0) for an example graph is shown below.

```
Enter the file name for the edges of the graph: graphEdges_2.txt
Enter number of nodes: 8

Predecessor List Entries
Predecessor of 0 : -1
Predecessor of 1 : 0
Predecessor of 2 : 1
Predecessor of 3 : 1
Predecessor of 4 : 2
Predecessor of 5 : 3
Predecessor of 6 : 4
Predecessor of 7 : 0

Shortest Paths from Vertex 0
0 - 1
0 - 1 - 2
0 - 1 - 3
0 - 1 - 2 - 4
0 - 1 - 3 - 5
0 - 1 - 2 - 4 - 6
0 - 7
```



Submission (as one Word or PDF document)
(1) The entire code, including the function added to the Graph class to print the shortest paths from the starting vertex (0) to the rest of the vertices in the graph.
(2) Do this manually: Draw a graph of 10 or more vertices, run the BFS algorithm on the graph (and identify the tree edges that connect a vertex to its predecessor vertex) and list down the predecessor for each vertex in the graph. The predecessor for the root vertex (0) is set as -1 and every other vertex should have a valid entry for its predecessor vertex in the BFS tree rooted at vertex 0. Using the tree edges, also list the shortest paths from vertex 0 to every other vertex in the graph (as shown in the sample screenshot above).
(3) Construct the edge list file for the test graph of 10 or more vertices created in part-2 and input the edge list file and the number of vertices to the code developed in part-1 and generate the output (for the predecessor list and the shortest paths from vertex 0) similar to the one shown in the above sample screenshot. The outputs in part-2 and part-3 are expected to match, if ties (arising while deciding to visit a neighbor vertex) are broken in favor of vertices with lower ID.