FLORIDA POLYTECHNIC UNIVERSITY

Points: 100/100 Due Date: Dec. 5th, 2021

Project #3: GraphMan

Submission Notes:

1. In addition to the code, please upload your report submission in PDF format. The report should have a clean and clear run of the system, including screenshots.

2. Do not submit any zip or compressed files, binary files, or any other generated files. Do not put the code inside the PDF file. Submission of unnecessary files or binary files in addition to source files will make you lose the points of the assignment. The code must have the following:
   a. Use of descriptive names for the identifiers and active names for functions.
   b. Use consistent indentation through your code.
   c. Use consistent braces style through your code.
   d. Efficient utilization of functions. Make sure to reduce (*keep functions as simple as you can make them*) and reuse (*create functions for repeated parts of the code*) your code.
   e. File-level comments, these comments will include the file name, your name, section number, and date of last modification. In addition, these comments must have instructions on how we run the program and test it.
   f. Function level comments include the function's description, the function's parameters, and the function's return value. These comments will appear before each of the functions, not the prototypes.
   g. In function comments, these comments will include a description of the atomic functionalities within the bodies of the functions.

Write a C++ program called GraphMan. The program reads a digraph from a graph file, performs a selected task, and displays the result. The graph file contains the number of vertices as its first line. The label list of the vertices starts at the second line. Following the label list is the list of edges. Each V_i, V_j, W line in the graph file represents an edge from vertex V_i to vertex V_j with a weight W for the edge. The program ignores any empty line in the graph file. The order of the edges per node should be the same as they appear in the graph file.
A menu should appear to the program user, and the user will choose the task that the program will perform on the digraph that is loaded from the file. The following is the list of tasks that the user can choose from:

- The shortest path between two nodes (using the weight)
- Print the adjacency list.
- Breadth-first search
- Depth-first search
- Depth-first search where the node edges are ordered depending on the increasing degree of the destination node

The graph ADT will have the following C++ classes:

- Graph
- Vertex
- Edge

In addition:

- An Edge class with the following:
    - Name.
    - Pointer to source vertex.
    - Pointer to destination vertex.
    - Weight
- A Vertex class with the following:
    - Name
    - A vector of pointers to the edges for which it is the source vertex.
- A vector that contains all the vertices, with their order.
- A vector that contains all the edges, with their order.
- Don't use C++ template in any of the classes.

An example for the content of graph file

```
4
a
b
c
d
a,b,5
a,c,3
b,d,2
c,d,4
d,a,3
```

Guidelines

1. Make sure to include sufficient documentation for your code. That includes comments and choosing the appropriate names for your data members, functions, classes, and objects.
2. Proper formatting of the code.
3. Separate the implementation and interface in all classes.