CSCI 2275 – Data Structures and Algorithms
Instructor: Archana Anand
Assignment 8
Due Friday, Nov 15, by 11:59 PM

People in Graphville

You are the mayor of Graphville. You have a list of all the people in your town. You also know the relation between them, basically if they know each other. You need to increase the community interaction between your constituents. Now before you do that you need to know a lot of other details like if two people know each other directly or through a friend or in any other way, also how many groups are there in your town.

What your program needs to do:

Build a graph. There is a file on canvas called people.txt that contains names of the people and list of names of people they know, similar to an adjacency list. When the user starts the program, read in the file and build a graph where every person is a vertex, and there is an edge to every person they know.

For this assignment, you are building the graph and implementing functionalities given in the menu.

Use a command-line argument to handle the filename.

Display a menu. Once the graph is built, your program should display a menu with the following options:

1. **Print list of people and their acquaintances**
2. **Print if people know each other**
3. **Print groups**
4. **Find the least number of introductions required**
5. **Quit**

Menu Items and their functionality:

Print list of people and their acquaintances. If the user selects this option, the vertices and adjacent vertices should be displayed.

Print if people know each other. This option takes two vertices as user inputs and displays True is the vertices are adjacent, and False if they are not adjacent.

If this is a part of the file:
Akhil-Paramvir,Aurangzeb
Aurangzeb-Akhil,Kieran
Earl-Paramvir,Kieran,Zack

Then Earl and Zack know each other
Kieran and Aurangzeb know each other
and so on.

Print groups. If the user selects this option, you need to do a depth-first search (DFS) of the graph to determine the connected people in the graph and assign those people the same group ID. The connected people are the vertices that are either directly connected by an edge, or indirectly through the series vertices. For example, in the above example, Akhil is connected to Kieran via Aurangzeb. When assigning group Ids, start at the first vertex and find all vertices connected to that vertex. This is group 1. Next, find the first vertex that is not visited yet. This vertex is the first member of group 2, and you can repeat the depth-first search to find all vertices connected to this vertex. Repeat this process until all vertices have been assigned to a group.

Printed Output Should look like below:

```
Group ID#1
Akhil
Aurangazeb
..
…

Group ID#2
..
...
```

Find the least number of introductions required- If the user selects this option, they should be prompted for the names of two people. Your code should perform BFS to find the shortest distance between two names. This is the minimum number of introduction that two people need. If there is no shortest distance, print “No way to introduce them”. You need to print the path if there is the shortest distance between nodes. For this, maintain a string variable called path in the struct

struct for each vertex is

```
struct vertex{
	string name;
	bool visited;
	int distance;
	string path;
	vector<adjVertex> adjacency;
	vertex(string n){
		name= n;
		visited = false;
		distance = 0;
	}
}
```

Every time you push the neighboring vertex of a node, you append to the name of the neighbor to the string path of the node separated by a comma.
For example.
If BFS starts from Akhil, then the path is Akhil
then when you push Paramvir, the path for vertex Paramvir is Akhil,Paramvir. All you need to do is append to the parent vertex’s path.

When you reach source vertex, the path of the parent gives the shortest path to reach the source

You may add any extra utility functions that you might need to implement the above functionality.

Submit it on .cpp file in Canvas