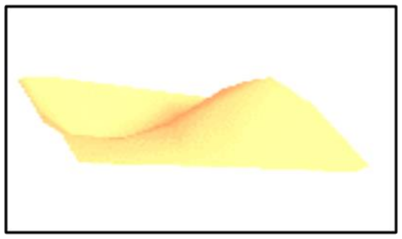
Assignment 4: File I/O and Containers: Constructing Connectivity Map among Different Cities in an Island

Description: In this assignment, you have to construct a graph showing the connectivity information among different cities in an island. There are 46 cities in the island. Please find attached a file named "**Terrain.obj**" that contains the topological information of the island. The terrain model of the island (below) shows that its topology is undulated.



In **Terrain.obj** file, lines starting with 'p' contains the 3-dimensional coordinates of the center of different cities. 'p' stands for position. The first line starting with 'p' refers to the city with index '0' and its corresponding center coordinates (x, y and z). Similarly, the second line starting with 'p' corresponds to the information for city 1 and so on. You will find there are information for 46 cities.

Next, you'll also find some lines starting with 'c' and followed by some integer. 'c' stands for connectivity information. Let us take the very first line of the file starting with character 'c':

```
c 13 28 41
```

This means cities 13, 28 and 41 are connected to each other.

Now, the next line shows more connection information:

```
c 13 41 42
```

That means city 13 has also connection with city 42 and vice versa, (41 repeated here);

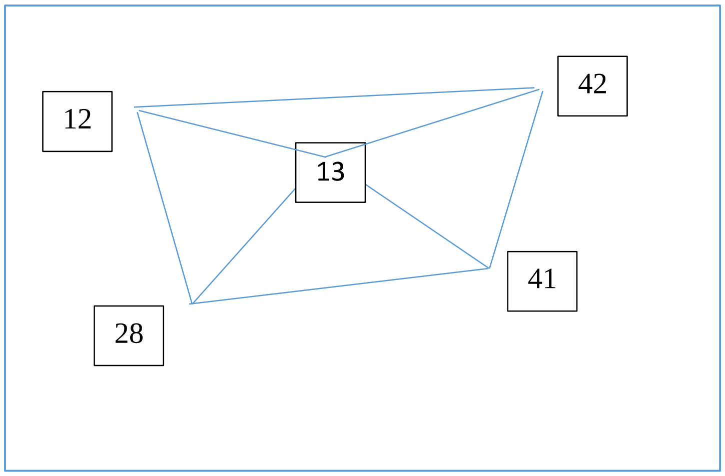
Down the road you'll see that city 13 has also connection with cities 12 and 28.

```
c 12 28 13
```

You'll also see some repeated information.

```
c 12 13 42
```

In the end, we can see that city 13 is connected to the following four cities: 12, 28, 41 and 42. Each line starting with character 'c' contains the connectivity information for 3 cities. City 13 has four connections and has formed 4 triangles with its 4 connectivity information.



You will find that there are seventy six connectivity information. We have identified connectivity information for city 13 from the above-mentioned four connectivity information. Similarly, you need to find out the connectivity information for all 46 cities (index 0 to index 45).

Construct a **Graph** class. You can consider the following member variables:

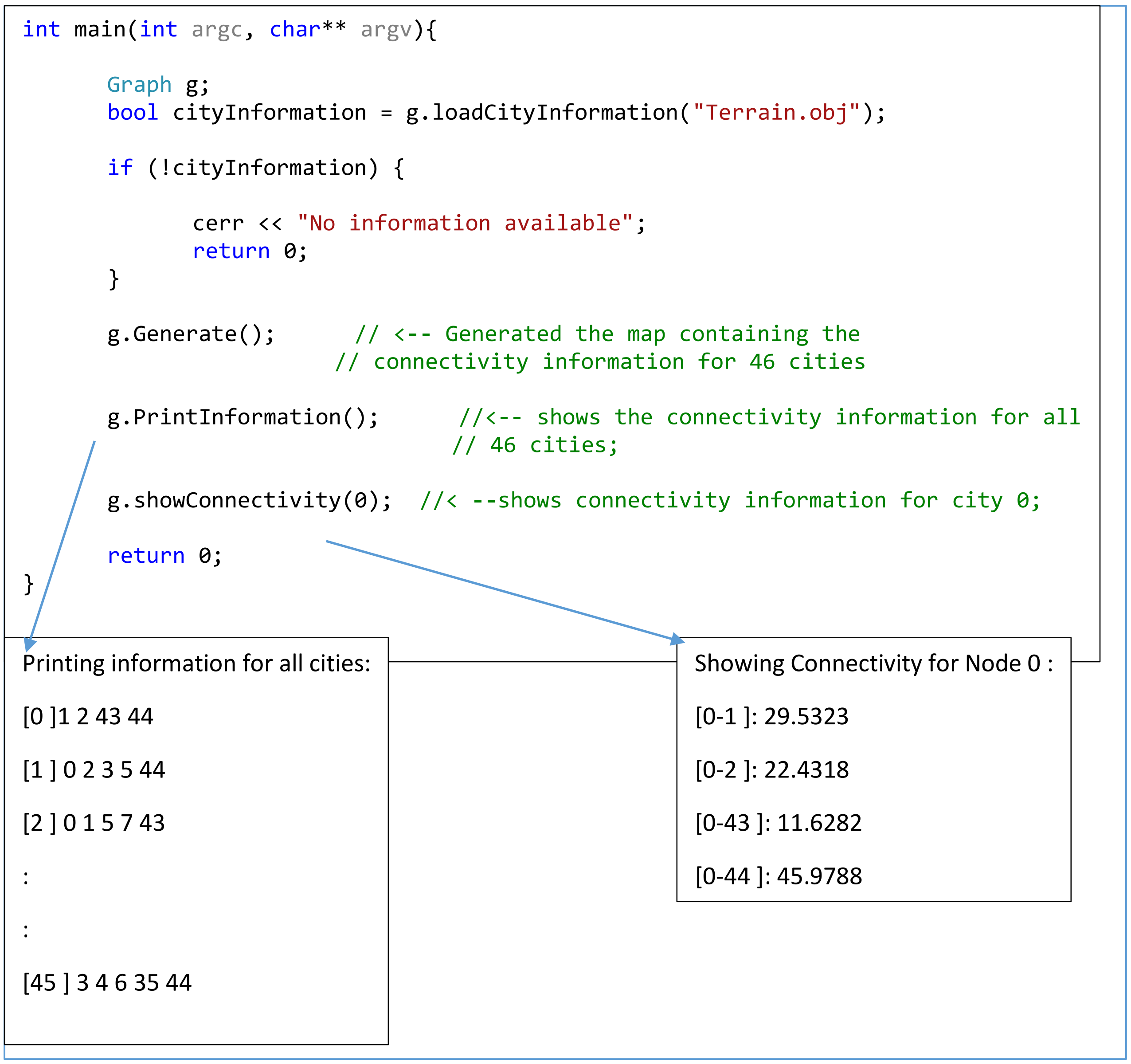
```
vector<Point3D> cityCoordinates;
vector<int> cityIndices;
map<int, set<int>> cityConnectivity;
```

You need to parse the file "**Terrain.obj**" to get information about center position (**cityCoordinates**) of different cities and their connectivity information (**cityIndices**) to other cities. Use **ifstream** and **istringstream** for extracting necessary information.

As you find, you also need to construct a **map** named **Connectivity** that will store connectivity information of all 46 cities.

Let us again consider the connectivity information of city 13. While constructing the map, you need to figure out the key and value pair. Here 13 is the key and the connected cities represented by a set are value {12, 28, 41, 42}. Set has been used to avoid duplicate city index. You need to construct the connectivity map for all 46 cities.

Take a look at the simple **main.cpp** file. Here there are some functions:



```
int main(int argc, char** argv){

        Graph g;
        bool cityInformation = g.loadCityInformation("Terrain.obj");

        if (!cityInformation) {

                cerr << "No information available";
                return 0;
        }

        g.Generate();       // <-- Generated the map containing the
                           // connectivity information for 46 cities

        g.PrintInformation();       //<-- shows the connectivity information for all
                                   // 46 cities;

        g.showConnectivity(0);  //< --shows connectivity information for city 0;

        return 0;
}
```

```
Printing information for all cities:

[0 ]1 2 43 44

[1 ] 0 2 3 5 44

[2 ] 0 1 5 7 43

:

:

[45 ] 3 4 6 35 44
```

```
Showing Connectivity for Node 0 :

[0-1 ]: 29.5323

[0-2 ]: 22.4318

[0-43 ]: 11.6282

[0-44 ]: 45.9788
```

The functions and the output are self-explanatory.

loadCityInformation takes the file as input, parse it and stores the necessary information in the container member variables. **Generate()** generates the connectivity map. **PrintInformation()** prints the {key, value} pair for all 46 cities. **ShowConnectivity()** takes a particular key and find the distances between the connected cities as shown above.

Submission:

You need to create at least two classes: **Point3D/ Vector3D** and **Graph**.

Compile and execute your code with g++ or the Microsoft compiler.

Place your solution in a zipped file named with your last name followed by the first initial of your first name followed by 4 (ex: **YasminS4.zip**) and submit the solution via canvas.

Include at the top of your solution (in comments) your name, what compiler(s) you used. Minimal documentation is required – use good naming conventions for variables and comment any non-trivial code (describe what it does). Finally, capture your output from program. If necessary, you can include a text file that describes your approach towards implementing this program.

Thus, your zip should contain the following:

- **Point3D.h/ Vector3D.h**, **Point3D.cpp/ Vector3D.cpp**, **Graph.h**, **Graph.cpp**. **graph_tester.cpp** and any additional file (if needed)
- output capture (name it something like **graph_tester_output**) and
- a text file describing your approach.

Submission deadline is **Friday, February 19, 11:59 pm**. Late submission will not be accepted. No excuse regarding mistake in uploading proper files will be considered.

This assignment carries a weightage of **15%** of the course.

Important Information:

Start assignment early! If assignment specification seems vague, please ask questions!!!

After grades are made available, any attitude towards creating chaos regarding vague assignment specification will not be excused. You may need clarification. Ask questions before the submission deadline.