CSC 210 HOMEWORK 4
REPRESENTATION OF GRAPHS

PROFESSOR GODFREY C. MUGANDA

1. Due Date

This assignment is due Wednesday of Week 7.

2. The Assignment

Write a Java class `GraphRepresentation` that represents a graph in the form of an adjacency list. An outline of the class is as follows

```
class GraphRepresentation
{
    // fields
    String[] vertexNames;
    List<List<Integer>> adjL;
    Map<String, Integer> strToIntMap;

    static GraphRepresentation getGraph(Scanner sc)
    {
        GraphRepresentation g = new GraphRepresentation();

        // missing code goes here

        return g;
    }

    @Override
    public String toString()
    {
        StringBuilder builder = new StringBuilder();

        // missing code goes here

        return builder.toString();
    }
}
```

Graph Data in the form of an adjacency list will be stored in a text file. The file will start out with an integer representing the number N of vertices in the graph. This will be followed by a list of N strings representing the names of the vertices. For example:

```
9
a b c d e f g h i
```

This data will be followed by N sets of data. Each set of data will correspond to the adjacency list for a single vertex. Each set of data will start out with a vertex v in the form of a string, followed by an integer, the out-degree $d(v)$ of v. This will be followed by $d(v)$ strings, the names of vertices that are reachable from v by a single edge. Continuing our example, the complete data for a graph will look like this

```
9
a b c d e f g h i
g 2 f h
h 1 i
a 2 b c
b 3 a c e
c  1 f
d 1 g
e 2 f i
f 4 d a b h
i 1 f
```

For example, the data set

```
g 2 f h
```

indicates that the vertex g has out-degree 2, and that the two edges that emanate from v have terminal vertices f and h.

The `GraphRepresentation` class will have a method

```
GraphRepresentation getGraph(Scanner sc)
```

that gets the above data from `Scanner` object linked to the input file and returns a `GraphRepresentation` object. Although the adjacency list information in the input file uses strings for vertex names, an internal representation will use integers for vertex names. The graph representation will establish an array

```
String[] vertexNames;
```

and a map

```
Map<String, Integer> strToIntMap;
```

to translate from the integer representations of vertex names to the string names of the vertices.

Finally, the adjacency list itself will be represented by a list of list of integers:

```
List<List<Integer>> adjL;
```

Alternatively, you can represent the adjacency list as an array of lists of integers. To do this, you have to define a type

```
class NeighborList extends ArrayList<Integer>
{
}
```

You can then use the declaration

```
NeighborList[] adjL;
```

Finally, you need to override the `toString()` method so return a string representation of the graph representation. The string returned should consist of the string representation of of the array, the map, and the adjacency lists in both the integer and string form. For the graph shown above, the string representation *must* look like this:

```
[a, b, c, d, e, f, g, h, i]
{a=0, b=1, c=2, d=3, e=4, f=5, g=6, h=7, i=8}

0 : [1, 2]
1 : [0, 2, 4]
2 : [5]
3 : [6]
4 : [5, 8]
5 : [3, 0, 1, 7]
6 : [5, 7]
7 : [8]
8 : [5]

a : [b, c]
b : [a, c, e]
c : [f]
d : [g]
e : [f, i]
f : [d, a, b, h]
g : [f, h]
h : [i]
i : [f]
```

Put your `GraphRepresentation` class in a file `GraphRepresentation.java`. You must use the following file, `GraphSearch.java` for your main class:

```
public class GraphSearch
{
    public static void main(String[] args) throws FileNotFoundException
    {
        JFileChooser chooser = new JFileChooser();
        FileNameExtensionFilter filter = new FileNameExtensionFilter(
                "Graph Data", "txt", "grf");
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(null);
        if (returnVal != JFileChooser.APPROVE_OPTION)
        {
            System.out.println("No file selected.");
            return;
        }
```

```
            File file = chooser.getSelectedFile();
            Scanner sc = new Scanner(file);

            GraphRepresentation graph = GraphRepresentation.getGraph(sc);
            System.out.println(graph);

        }
    }
```

To test your program, create a file with either a `.txt` or `.grf` extension and store it in the `Documents` folder of your Windows machine, or your home folder on your Mac. If you place the test file any other place, you will have to use the file chooser to navigate to the location containing the file.