CS4115 Substitute Project: Graph Algorithms
Programming Assignment
DueDate: 18.00, Wed, ~~Week14 (6th May)~~ Week15 (13th May),

Objective: The objective of this project is to demonstrate understanding of, and competence using, graph algorithms as discussed in lectures. On a data set you have already seen that represents road distances you will you will be asked to answer some initially basic graph queries culminating in measuring the distance between two specified nodes.

I will run your program twice: firstly on a small dataset and then on a very large dataset that will test the efficiency of your choice of data structures and algorithms. The assignment counts for 90% of your semester grade so it will be marked out of 90. For reference the marking scheme is given below.

Category	Query	Marks
File submitted		5
Clean compilation		15
Order and Size (n, m)	1	10
Degree:		
- Largest	2	5
- Average	3	5
Neighbors:		
- edge-distance 1	4	5
- edge-distance k	5	15
Distance:		
- Shortest Path	6	15
- Shortest Edge Count	7	15
Total		90

Table 1: Marking scheme for assignment.

Your job: The road data will be given to you in two files, `country.osm.graph` and `country.osm.xyz`. These will be exactly as the format of those referred to in Lab06's write-up. With these data you should answer the questions above.

Data Format: The data in the `.xyz` file comprises x-y coordinates[1] where the i^{th} line represents the coordinates of vertex i. After comment lines at the top of `country.osm.graph`

[1] Actually, z values, too, but you can ignore it.

the first line of useful data specifies the number of nodes and edges in the road network. The remainder of that file is made up of edge lists: one for each vertex, in the order 1 to n. Note that some vertices might have no neighbors.

The data was pulled from OpenStreetMap – hence the `.osm` in the file names – and, they appear to be all undirected. That is, both of the edges (u, v) and (v, u) are listed, once on u's adjacency list and again, later, on v's.

You should represent your graph in the *formatz* representation. That is, instead of thinking of representing the graph as n $n \times n$-matrix where almost all of the entries are unused, use *formatz*. Each vertex can store its neighbors as a linked list and since every vertex will have *some* neighbors you can do this with a vector. So the graph is represented as a vector of linked lists *à la* formatz.

You should then open for reading the files `country.osm.graph` and `country.osm.xyz` where `country` will have been read from standard input (the keyboard). Be prepared for this string to be a file path such as something like `/home/cs4115/labs/lab06/italy`, to which you would append `.osm.graph` and `.osm.xyz`.

Graph Queries: Having read the graph and represented it internally you are now ready to answer the above questions. Some queries require input, either a vertex (for the two "Neighbors" and "Distance" queries) and an additional value k for collecting vertices edge-distance k away. Read these as `int`s from stdin as you need them. So the input to the program could be

```
/home/cs4115/labs/lab06/italy
34      # vertex 34
27 4    # nbrs of vertex 27, 4 hops away
29 11   # SP(29,11)
38 6    # SP(38,6) but by edges, not xy dist
```

Some points regarding the expected output. You should

- answer them in the exact order you see above with no directions from the user
- no extraneous output text[2], just the facts
- for outputs requiring floating point values (average degree and shortest path are the only two) please round them to 6 decimal places
- for outputs requiring lists (neighbors, etc.) please order the output smallest to largest – sorry, `handin` again.

Useful Resources: Very soon I will make available my solutions to the outstanding labs as you could draw on these. I will also post sample input / output for comparison.
Good luck.

[2]I'm hoping to automate the grading of these using something akin to `handin`.