HW3: Geospatial data handling

Total points: 6

In this homework, you are going to work with **spatial data** - you will collect/create (generate/sample) your own spatial data, visualize it, do queries on it, and visualize the query results.. **Hope you have fun with this!** Geospatial data handling is a valuable skill!

The exercise will give you a taste of working with spatial data, use of a spatial file format and spatial query functions, all of which are quite useful from a real-world (or job interview) perspective.

What you need to do is described below in sufficient, but not too much, detail - you'd need to do a bit of reading up and experimenting, to fill in the gaps. Please post on Piazza, or talk to a TA/CP, or me, if you are unable to proceed at any point!

1. You need to create (generate) latitude,longitude pairs (ie. spatial coordinates) for **12 locations** on the USC campus, in addition to where your **home/apartment/dorm room** is. The 12, other than your home, would have to be spread out - use your judgment, try to span the campus area, choose locations inside and outside, nearby - we don't want to cover a 'huge' region, or at the other extreme, sample just parts of a single building! If you are not on campus (if you are a DEN student, live abroad/away from campus), feel free to substitute places around where you live, for the 12 points other than your **residence or work** (again, make sure they are not too close to each other or too far apart). The 12 points need to be in these 4 categories, 3 points per category: campus entrances (gates), libraries, waterworks, eateries :)

How would you obtain (lat,long) spatial coordinates at a location? By walking around!! Please do NOT click on https://www.google.com/maps to acquire your data! You need to submit a selfie of you [if you don't want

to show your full face, that's ok, include just a part :)] at each location, to show that you were there. Bring up this page on your phone, it will display your location - write it down/type it up, along with the location's name; to get (long,lat), you can also your phone's compass, or a downloaded app.

Make a simple (text/on paper) table of locations+names, 13 entries total, as a starting point for what follows.

2. Now that you have 13 coordinates and their label strings (ie. text descriptions such as "Tommy Trojan", "SAL", "Chipotle"..), you are going to create a KML file (.kml format, which is XML) out of them using a text editor. Specifically, each location will be a 'placemark' in your .kml file (with a label, and coords). Here is more detail. The .kml file with the 13 placemarks is going to be your starter file, for doing visualizations and queries. Here is a .kml skeleton to get you started (just download, rename and edit it to put in your coords and labels). NOTE - keep your labels to be 15 characters or less (including spaces). Here is the same .kml skeleton in .txt format, if you'd like to RMB save it instead and rename the file extension from .txt to .xml. NOTE too that in .kml, you specify (long,lat), instead of the expected (lat,long) [after all, longtitude is what corresponds to 'x', and latitude, to 'y']! Also, place your 12 coords (not your home one) in KML 'folders'.

You are going to use Google Earth to visualize the data in your KML file (see #3 below). FYI, as a quick check, you can also visualize it using this page - simply copy and paste your KML data into the textbox on the left, and click 'Show it on the map' to have it be displayed on a map on the right :) Or you can do a quick check at http://kmlviewer.nsspot.net/ instead.

3. Download Google Earth on your laptop, install it, bring it up. Load your .kml file into it - that should show you your sampled locations, on Google Earth's globe :) Take a snapshot (screengrab) of this, for submitting.

4. Install Oracle 11g+Oracle Spatial, or Postgres+PostGIS on your laptop, and browse the docs for the spatial functions and search for basic tutorials (eg. this is a good one).

Windows users: please install Postgres v.13.3 from here, then install PostGIS v.3.1.2 after starting Postgres' Stack Builder UI. Mac users can do this instead.

After you install Postgres+PostGIS, here is how you can create a DB, create a table, insert data, query it.

5. You will use the spatial db software to execute the following two spatial queries that you'll write:

• **compute the convex hull** for your 13 points [a convex hull for a set of 2D points is the smallest convex polygon that contains the point set]. If you use Oracle, see this page; if you decide to use Postgres, read this and this instead. Use the query's result polygon's coords, to create a polygon in your .kml file (edit the .kml file, add relevant XML to specify the KML polygon's coords). Load this into Google Earth, visually verify that all your points are on/inside the convex hull, then take a screenshot. Note that even your data points happen to have a concave perimeter and/or happen to be self-intersecting, the convex hull, by definition, would be a tight, enclosing boundary (hull) that is a simple convex polygon. The convex hull is a very useful object - eg. see this discussion.. Note: be sure to specify your polygon's coords as '...-118,34 -118,34.1...' for example, and not '...-118, 34 -118, 34.1...' [in other words, do not separate long,lat with a space after the comma, ie it can't be long, lat].

• **compute the four nearest neighbors** of your home/apt/dormroom location [look up the spatial function of your DB, to do this]. Use the query's results, to create four line segments in your .kml file: line(home,neighbor1), line(home,neighbor2), line(home,neighbor3), line(home,neighbor4). Verify this looks correct, using Google Earth, take a snapshot.

Note - it *is* OK to hardcode points, in the above queries! Or, you can create and use a table to store your 13 points in it, then write queries against the table.

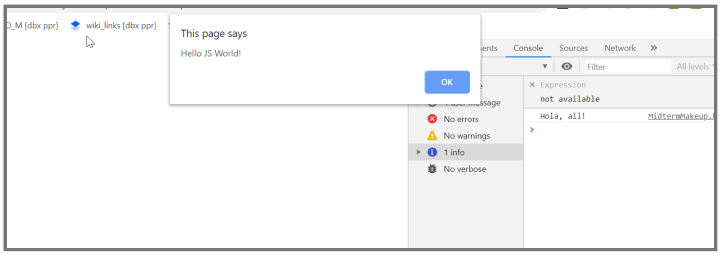
6. Use OpenLayers (a JavaScript API) to visualize your location data. The idea is to store your 13 sampled points, via your web browser, in a browser cache area in your local machine, where the data would persist [even after you close the browser]; then you'd read back the stored

values, and visualize them, using the OpenLayers API. To store and load points, you'll use 'HTML5 localStorage', which is a key-value based standard WWW API.

The API calls are in JavaScript, which you would run via an html page, like so:

```
<!DOCTYPE html>

<html>

<head>
<title>OL</title>
</head>

<body>
<script>
// your JS code
alert("Hello JS World!");
console.log("Hola, all!");
</script>
</body>

</html>
```

If you create an html file [OL.html] and run it (in Chrome, preferably), you'd see this (in Chrome, to see the console, use the three-dots dropdown menu on the top right: More tools -> Developer tools) :



Now you're ready to store, retrieve, and plot your points!

The code in this starter OL.html is to show you, roughly, how to do it; you'd need to do it 'for real' by modifying it :) EVEN IF you don't know JS or coding, don't worry - it's easy, and fun, to figure it out! In other words, start with the OL.html I'm giving you, mildly EDIT it to ADD IN your sampled locations, submit that [note: you **do NOT** have to do extensive JavaScript coding at all!]

If you like, you can do the location visualization, using CodePen [https://codepen.io/] or jsfiddle [https://jsfiddle.net/], and just submit a URL of your code - the grader would simply copy and paste your URL into their browser, and be able to see your code and the result. If you do this, submit a README file with your link, instead of submitting OL.html.

7. Using Tommy Trojan as the center, **compute** (don't/can't use GPS!) a set (sequence) of lat-long (ie. spatial) co-ordinates that lie along a pretty **Spirograph**™ curve :)

If you are on campus, you can go visit TT to get his (long,lat); otherwise, please long-press (for just under a second) on him in https://www.google.com/maps, that will make his coords pop up in a little box.

Create a new KML file with Spirograph curve points [see below], convert the KML to an ESRI 'shapefile', visualize the shapefile data using ArcGIS Online.

To convert your .kml into a shapefile, use this online converter: https://mygeodata.cloud/converter/kml-to-shp - the result will be a .zip [which is what we call 'shapefile'], which will contain within it, shape data (.shp), a relational table (.dbf), and other optional files (.shx, .prj, .cpg). Here is a page on shapefiles, and this talks about the various components (.shp, .dbf etc) of a shapefile.

Once you have your shapefile, you can upload it to ArcGIS' online map creator to view your Spirograph curve-shaped points. To do so, log on to ArcGIS [after creating a free 'public' account], at https://www.arcgis.com/, then use the 'Map' tab - https://www.arcgis.com/home/webmap/viewer.html?useExisting=1. Do 'Add -> Add Layer from File', and upload your shapefile .zip, you should see your data overlaid on a map. Here is a screenshot of roughly what to expect.

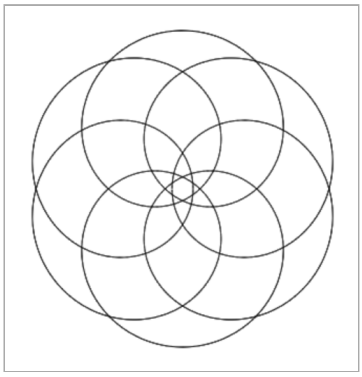
What if your shapefile is too large, and you get a 'Dataset too large' error while using ArcGIS? In that case, you can import your shapefile into https://maps.equatorstudios.com/ instead, and create a screenshot (this

site will handle large shapefiles).

For the Spirograph curve point creation, use the following parametric equations (with R=6, r=1, a=8):

```
x(t) = (R+r)*cos((r/R)*t) - a*cos((1+r/R)*t)
y(t) = (R+r)*sin((r/R)*t) - a*sin((1+r/R)*t)
```

Using the above equations, loop through t from 0.00 to n*Pi (eg. 2*Pi; note that 'n' might need to be more than 2, for the curve to close on itself; and, t is in radians, not degrees), in steps of 0.01. That will give you the sequence of (x,y) points that make up the Spiro curve, which would/should look like the curve in the screengrab below, when R=6, r=1, a=8:



Note - your figure MUST resemble the above, ie. it MUST have 6 loops - so please do not change R, r or a!

In order to center the Spirograph at a given location [TT or other], you need to ADD each (x,y) curve point to the (lat,long) of the centering location - that will give you valid Spiro-based spatial coords for use in your .kml file. You can use any coding language you want, to generate (and visualize) the curve's coords: JavaScript, C/C++, Java, Python, SQL, MATLAB, Scala, Haskell, Ruby, R.. You can also use Excel, SAS, SPSS, JMP etc., for computing [and plotting, if you want to check the results visually] the Spirograph curve points. This can help you a lot :)

Payoff - when you do this in KML, what you'll see is the Spirograph curve superposed on the land imagery - pretty!

PS: Here is MUCH more on Spirograph (hypocycloid and epicycloid) curves if you are curious. Also, for fun, try changing any of R, r, a in the code for the equations above [you don't need to submit the results]!

Here is what you need to **submit** (**as a single .zip file**):

* 13 selfies, from step 1 above [if you don't submit these, you will LOSE 2 points!]

* your .kml file from step 5 above - with the placemarks, convex hull and nearest-neighbor line segments (**1 point**)

* a text file (.txt or .sql) with your two queries from step 5 - table creation commands (if you use Postgres and directly specify points in your queries, you won't have table creation commands, in which case you wouldn't need to worry about this part), and the queries themselves (**2 points**)

* screengrabs from steps 3,5 (**1 point**)

* a .html file (with the OpenLayers code) from step 6, or a CodePen/jsfiddle link (**1 point**)

* your Spirograph point generation code, the resulting .kml file ("spiro.kml"), shapefile (this needs to be a .zip) and a screenshot (**1 point**)

HAVE FUN! From here on out, you know how to create custom overlays (via KML files containing vector symbols constructed from points, lines and polygons, and its shapefile equivalent) on a map, and perform spatial queries on the underlying data. This can even be the basis for a career :)
