HW3: assigned 3/14, due 3/29 before 11:59 PM

Total points: 6 [plus 1 bonus]

In this homework, you are going to work with **spatial data** - you will create (generate/sample) some data visualize it, do queries on it, and visualize the query results.. Hope you have fun with this!

The exercise will give you a taste of working with spatial data, use of a spatial file format and spatial que 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 c reading up and experimenting, to fill in the gaps. Please talk to a TA/grader if you are unable to proceed any point!

1. You need to create (generate) latitude,longitude pairs (ie. spatial coordinates) for **9 locations**. One of those will be where your home/apartment/dorm room is. The other eight would have to be spread out - least 100 feet between adjacent locations (and at most 200 or 300 or even 400 feet between neighbori points) - we don't want to cover a 'huge' region, or at the other extreme, sample just parts of a single building!

If you are on campus, you can obtain the coords of the four corners (Exposition/Vermont, Vermont/Jefferson, Jefferson/Figueroa, Figueroa/Exposition), and get coordinates for four spots inside tl campus (classrooms, labs, offices, restaurants, landmarks..). If you are a DEN student, please get your 8 coordinates from your place of work or your home neighborhood (again, make sure they are not too clc to each other or too far apart).

How would you obtain (lat,long) spatial coordinates at a location? You can do so, one of two ways:

- **using the Chrome browser**, simply bring up this (geolocate_mod/geolocate_mod.html) page on your smartphone (that has GPS), and writ down the (latitude,longitude) values that get shown when you load/refresh the page :) As you can see, the page shows your location on a m cool! Be sure to enable cross-site script loading when you run this (because the script is on our Dropbox area, and accesses a map API at google.com) - click on the shield icon at the right of the URL bar, and click on 'Load unsafe scripts'. Alternately, you can use this (geoloc2/run.html) page to obtain the (latitude,longitude) coordinates.
- **using your phone's built-in GPS/compass app**, simply read off the displayed GPS coordinate values (if the coordinate display is in degrees, minutes and seconds, you need to convert the minutes,seconds pair of values into a single fractional degree value - one degree is subdivided 60 minutes (60'), and one minute is subdivided into 60 seconds (60'') - so for example, 30'15", since it is equivalent to 1815", would be eqv 1815/3600=0.504 degrees.

Also, be sure to write down the location names as well (you will use them to label your points when displaying). AND, take a selfie (!) that clearly shows the location you're sampling - **this step is to ensure t you're not simply reading off the coords from a map, sitting at home**!

2. Now that you have 9 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). He (https://developers.google.com/kml/documentation/kml_tut#placemarks) is more detail. The .kml file w the 9 placemarks is going to be your starter file, for doing visualizations and queries. Here (data/starter_kml.xml) 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).

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 (http://display-kml.appspot.com/) 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 m on the right :)

3. Download Google Earth (https://www.google.com/earth/download/ge/agree.html) on your laptop, install it, bring it up. Load your .kml file into it - that should show you your 9 sampled locations, on Goo 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. Here (BigSQL/index.html) is my page that walks you through installing a packaged version of Postgres.

4 (alt). You can also use MySQL (https://dev.mysql.com/doc/refman/5.7/en/spatial-extensions.html) if yo want, or even sqlite (http://www.bostongis.com/PrinterFriendly.aspx?content_name=spatialite_tut01); you are familiar with using SQL Server (https://docs.microsoft.com/en-us/sql/relational-databases/spatial/spatial-data-sql-server), that can also help you do the homework. Even QGIS (https://gis.stackexchange.com/questions/38937/how-to-connect-to-postgres-with-qgis) can be used to do the HW.

4 (alt alt). IF YOU ARE FEELING ADVENTUROUS: as an alternative to installing Oracle or Postgres (or MyS or sqlite or SQL Server...) on your machine, you can use Postgres or Oracle on the AWS cloud platform (i without installing anything on your laptop!) - eg. see this (https://aws.amazon.com/free/?) page, and th (https://aws.amazon.com/rds/postgresql/) one. **Be sure to not leave your DB instance running, when yo aren't working on the hw!**

4 (alt alt alt). Last but not least, do feel free to use GCP for this! You are already set up to do GCP-basec data processing, so why not? :) Here are some relevant resources:

* https://cloud.google.com/sql/docs/postgres/quickstart (https://cloud.google.com/sql/docs/postgres/quickstart)

* https://medium.com/google-cloud/postgres-is-incredibly-awesome-c54353b88655 (https://medium.com/google-cloud/postgres-is-incredi awesome-c54353b88655)

* https://cloudplatform.googleblog.com/2017/03/Cloud-SQL-for-PostgreSQL-managed-PostgreSQL-for-your-mobile-and-geospatial-applications-in-Google-Cloud.html (https://cloudplatform.googleblog.com/2017/03/Cloud-SQL-for-PostgreSQL-managed-PostgreSQL-for-yo mobile-and-geospatial-applications-in-Google-Cloud.html)

* https://cloudplatform.googleblog.com/2017/08/Cloud-SQL-for-PostgreSQL-updated-with-new-extensions.html (https://cloudplatform.googleblog.com/2017/08/Cloud-SQL-for-PostgreSQL-updated-with-new-extensions.html)

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 9 points [a convex hull (http://mathworld.wolfram.com/ConvexHull.html) for a set of 2D points is the smallest convex polygon that contains the point set]. If you use Oracle, see this (https://docs.oracle.com/cd/A97630_01/appdev.920/a96630/sdo_aggr.htm) page; if you decide to use Postgres, read this (http://postgis.net/docs/ST_ConvexHull.html) and this (http://stackoverflow.com/questions/10461179/k-nearest-neighbor-query-in-postgis) 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 your 9 points are inside the convex hull, then take a screenshot. Note that 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 (https://www.quora.com/Wha are-the-real-life-applications-of-convex-hulls) discussion.. If you want to explore geometry algorithms (of which convex hull computation is o in more detail, this (http://geomalgorithms.com/algorithms.html) is a great resource [thanks to Mark Debord (a student from a previous offe of the course) for the link].

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

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

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

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

* your 9 selfie pics that 'prove' you actually collected the point locations on site (.jpg or .png) (no points for selfies submission, but, -2 points YOU DON'T SUBMIT ALL NINE)

* 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+2=4 points)

* screengrabs from steps 3,5 (1 point)

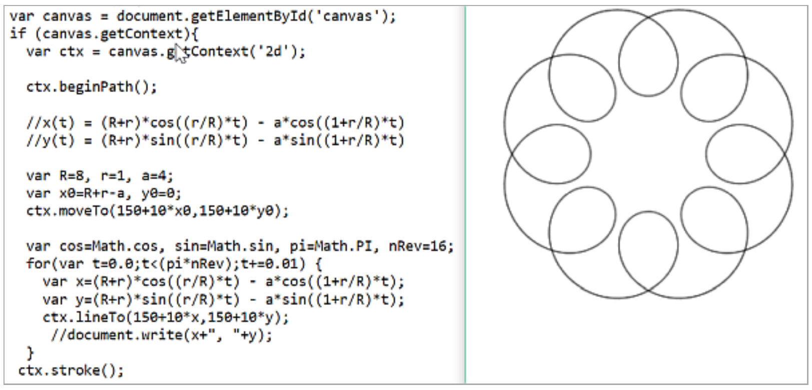
BONUS QUESTION! [1 point]

Using SGM 123 as the center, **compute** (don't use GPS!) a set (sequence) of lat-long (ie. spatial) co-ordinates that lie along a pretty Spirograph(TM) curve (https://www.google.com/search?q=Spirograph+curve&num=100&source=lnms&tbm=isch) :) Create a new KML file with these points, visualize it on Google Earth, submit these three items: your point generation code (see below), the resulting .kml file ("spiro.kml") and a screenshot ("spiro.jpg" or "spiro.png"). DEN students: you can use the center, a different spatial coordinate (eg. that of your home).

For the Spirograph curve point creation, use the following parametric equations (with R=8, r=1, a=4):

```
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 cu in the right side of the screengrab below, when R=8, r=1, a=4 (my JavaScript code for the point generation+plotting loop is on the left):



```
var canvas = document.getElementById('canvas');
if (canvas.getContext){
  var ctx = canvas.getContext('2d');

  ctx.beginPath();

  //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)

  var R=8, r=1, a=4;
  var x0=R+r-a, y0=0;
  ctx.moveTo(150+10*x0,150+10*y0);

  var cos=Math.cos, sin=Math.sin, pi=Math.PI, nRev=16;
  for(var t=0.0;t<(pi*nRev);t+=0.01) {
    var x=(R+r)*cos((r/R)*t) - a*cos((1+r/R)*t);
    var y=(R+r)*sin((r/R)*t) - a*sin((1+r/R)*t);
    ctx.lineTo(150+10*x,150+10*y);
     //document.write(x+", "+y);
  }
 ctx.stroke();
```

In order to center the Spirograph at a given location [SGM123 or other], you need to ADD each (x,y) cu point to the (lat,long) of the centering location - that will give you valid Spiro-based spatial coords for us in your .kml file. You can use any coding language you want, to generate (and visualize) the curve's coo JavaScript, C/C++, Java, Python, SQL (https://docs.oracle.com/cd/B28359_01/server.111/b28285/sqlqr02.htm), MATLAB, Scala, Haskell, Rub 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.

What you'll see is the Spirograph curve, superposed on the land imagery - pretty!

PS: Here (https://www.google.com/search?q=Spirograph+curve&ie=utf-8&oe=utf-8) is MUCH more on Spirograph (hypocycloid and epicycloid) curves if you are curious. Also, for fun, try changing any of R, r, in the code for the equations above [you don't need to submit the results]!

HAVE FUN! From here on out, you know how to create graphics (KML files containing vector symbols constructed from points, lines and polygons) overlaid on any map, and perform spatial queries on the underlying data :)
