HW2: Assigned 10/22, due 11/3 by 11:59 PM

Total points: 2+6+2=10 [read below for the breakdown] [plus 1 point extra credit!]

In this homework, you are going to work with **spatial data** - you will create 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 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 talk to a TA/grader if you are unable to proceed at any point!

1. You need to create (generate) [latitude,longitude] spatial coordinates for 10 locations. One of those needs to be where your home/apartment/dorm room is. The other 9 would have to be spread out - spatially distinct, at least 100 feet between adjacent locations (and at most 'several hundred feet' - we don't want to cover a huge region overall!). If you are on campus, you can obtain the coords of its four corners (Exposition/Vermont, Vermont/Jefferson, Jefferson/Figueroa, Figueroa/Exposition), and get coordinates for 5 spots inside the campus (classrooms, labs, offices, restaurants, landmarks..). If you are a DEN student, get your coordinates from your place of work or neighborhood (again, make sure they are not too close to each other or too far apart).

How would you obtain spatial coordinates at a location? You can do so one of two ways:

* **using the Chrome browser**, simply bring up [this](https://dl.dropboxusercontent.com/u/91263185/!shared/courses/CS585/f16_DSRDB/hw/HW2/geolocate_mod/geolocate_mod.html) page on your smartphone (that has GPS), and write 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 map - 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](https://dl.dropboxusercontent.com/u/91263185/!shared/courses/CS585/f16_DSRDB/hw/HW2/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 into 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 eqvt to 1815/3600=0.504 degrees.

Note that in LA, you will (should!) get coordinates similar to [this.](http://www.latlong.net/place/los-angeles-ca-usa-1531.html)

Also, be sure to make a note of the location names as well (you will use them to label your points when displaying them on a map).

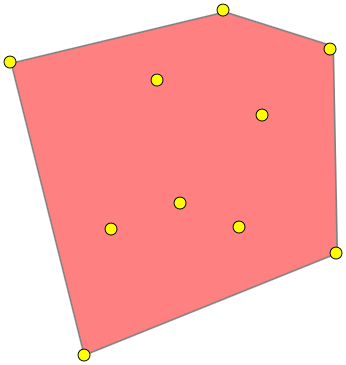
2. Now that you have 10 coordinates and their label strings (ie. text descriptions such as "Tommy Trojan", "SAL", "Chipotle"..), you are going to create a KML file (.kml format) out of them using a text editor. KML is a map-oriented file format, with XML tags. Specifically, each location you surveyed will be a 'placemark' in your .kml file (specified using coords and labels). [Here](https://developers.google.com/kml/documentation/kml_tut" \l "placemarks)is more detail. The .kml file with the 10 placemarks is going to be your starter file, for doing visualizations and queries. [Here](https://dl.dropboxusercontent.com/u/91263185/!shared/courses/CS585/f16_DSRDB/hw/HW2/data/starter.kml) is a .kml skeleton to get you started (just download, rename and edit it to put in your coords and labels). NOTE - keep each label to be 15 characters or less (including spaces) - otherwise they might not be displayed properly.

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 10 sampled locations, on Google Earth's globe :) Take a snapshot (screengrab) of this, for submitting.

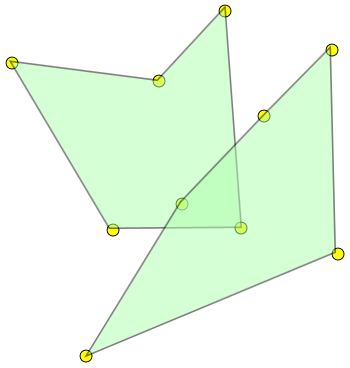
4. Install Oracle 11g+Oracle Spatial, or Postgres+PostGIS on your laptop, and browse the docs for the spatial functions - you can use either of these for query part of the assignment. Feel free to search stackoverflow for [PostGIS](http://stackoverflow.com/questions/tagged/postgis)questions, and [Oracle Spatial](http://stackoverflow.com/search?q=oracle+spatial) questions, to discover how spatial queries are written. Do spend some time on getting to know the API calls! If you can't get Postgres/PostGIS to install properly on your Mac or PC, you could consider using it via Amazon's AWS - see [this](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.PostgreSQL.CommonDBATasks.html#Appendix.PostgreSQL.CommonDBATasks.PostGIS) note. You can also use MySQL, sqlite etc. if you want, you're not limited to using Oracle or Postgres.

You need to use the above software to execute the following two spatial queries that you will need to write:

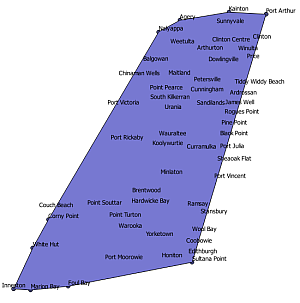
\* **write a query to compute the convex hull for your 10 points** [a [convex hull](http://mathworld.wolfram.com/ConvexHull.html) for a set of 2D points is the smallest convex polygon that contains the point set]. If you use Oracle, see [this](https://docs.oracle.com/cd/A97630_01/appdev.920/a96630/sdo_aggr.htm) page; if you decide to use Postgres, read [this](http://postgis.net/docs/ST_ConvexHull.html) page instead. Using the convex hull query's result polygon's coords, create a polygon in your .kml file (edit the .kml file, add relevant XML to specify the polygon's coords). Load the resulting .kml into Google Earth, visually verify that your 10 points are bounded by the convex hull that your query returned, then take a screenshot. Conceptually, the convex hull of a set of 10 points would look like this:



\* assuming the points (your collected locations) are called #1,#2,#3....#10, create a polygon using your points #1,#2,#8,#9,#10 (in that order), and another polygon with the remaining points in order (#3,#4,#5,#6,#7). Then **write a query to find out if the two polygons overlap** - the result would be (Boolean) true or false, depending on your coordinates. Add these two polygons to your .kml file, and visually verify (in Google Earth) the overlap as being true or false. Take a screenshot. Conceptually, the pair of polys from the same set of 10 points from above, could look like so [in this case they overlap; with your own data, they might not]:

   
UPDATE: feel free to REARRANGE the points #1,#2,#8,#9,#10 to get a non self-crossing polygon (if you get a self-crossing polygon when you don't reorder and that bothers you!); likewise feel free to reorder #3 through #7. Doing so might give you a *different* result for the overlap [compared to the result from the polygons where the points are all in ascending order], which is fine.

Be sure to show your locations' text labels (over satellite view or map view), and be sure to make your convex hull and the two other polygons translucent (via opacity values). Eg. a translucent convex hull would look like this:



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

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

\* your .kml file from step 4 above - with the placemarks, convex hull and two region polygons (2 points)

\* a text file (.txt or .sql) with your two queries from step 4 - 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 two queries themselves (2+2+2=6 points)

\* screengrabs from steps 3,4 (1+1=2 points)

Extra credit: 1 point

Please submit a separate screenshot for this..

Show a ['deltoid' curve](https://en.wikipedia.org/wiki/Deltoid_curve) overlaid on the USC campus - use SGM123 as the origin for the curve, calculate points on the curve (see equation below), and create a KML file with the SGM123 coords and the deltoid coords, view it in Google Earth, take a snapshot, submit :)

Here is the deltoid curve:

x = 2a.cos(phi) + a.cos(2.phi)   
y = 2a.sin(phi) - a.sin(2.phi)

As you can see, the above is a parameteric curve, with the free parameter phi - it varies from 0 to 2pi as it sweeps out the curve. 'a' is a scaling constant that you can play with (eg. start with 1.0, and scale it *down*, eg. to 0.01) - after you see what different values of 'a' do to the result, set it so that the deltoid curve roughly spans the USC campus. Note - your (x,y) coords from the curve formulae would need to be 'translated' so that the curve is centered at SGM123 as mentioned above (add tX,tY to each point, where tX,tY is SGM 123's coordinate). Find the coords for SGM123 using the method you used to get your 10 coords for the main part of the HW; if you're a DEN student, ask a TA/grader :)

For generating the coordinates for the curve, feel free to use any language you want: C++, JavaScript, Java, Python, C#, SQL (!), R.. FYI I did [mine](https://dl.dropboxusercontent.com/u/91263185/!shared/courses/CS585/f16_DSRDB/hw/HW2/pics/my_deltoid.png) in JavaScript.

Here is one possible result, showing the deltoid curve centered at our classroom:



HAVE **FUN**! This homework shows how we can conceptualize and visualize place/space using free digital tools, APIs and formats! From here on out you know how to create map graphics for your own purposes (KML files containing spatial vector symbols constructed from points, lines and polygons) :) You could even wrap all this into apps and make them available to others..